UNITED STATES AIR FORCE

SUMMER RESEARCH PROGRAM -- 1999

SUMMER RESEARCH EXTENSION PROGRAM FINAL REPORTS

VOLUME 3

ROME LABORATORY

RESEARCH & DEVELOPMENT LABORATORIES

5800 Uplander Way

Culver City, CA 90230-6608

Program Director, RDL
Gary Moore

Program Manager, AFOSR
Colonel Jan Cerveny

Program Manager, RDL
Scott Licoscos

Program Administrator, RDL
Johnetta Thompson

Program Administrator, RDL
Rebecca Kelly-Clemmons

Submitted to:

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

Bolling Air Force Base

Washington, D.C.

December 1999

PREFACE

This volume is part of a four-volume set that summarizes the research of participants in the 1999 AFOSR Summer Research Extension Program (SREP). The current volume, Volume 1 of 4, presents the final reports of SREP participants at Armstrong Laboratory.

Reports presented in this volume are arranged alphabetically by author and are numbered consecutively -- e.g., 1-1, 1-2, 1-3; 2-1, 2-2, 2-3, with each series of reports preceded by a 35 page management summary. Reports in the four-volume set are organized as follows:

VOLUME	TITLE
1	Armstrong Research Laboratory
2	Phillips Research Laboratory
3	Rome Research Laboratory
4	Wright Research Laboratory

REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-00-

0721

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December, 1999	**3. REP**
4. TITLE AND SUBTITLE 1999 Summer Research Program (SRP), Summer Research Extension Program (SREP), Final Report, Volume 3, Rome Laboratory		**5. FUNDING NUMBERS** F49620-93-C-0063
6. AUTHOR(S) Gary Moore		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Research & Development Laboratories (RDL) 5800 Uplander Way Culver City, CA 90230-6608		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research (AFOSR) 801 N. Randolph St. Arlington, VA 22203-1977		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES		
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release		**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

The United States Air Force Summer Research Program (SRP) is designed to introduce university, college, and technical institute faculty members to Air Force research. This is accomplished by the faculty members, graduate students, and high school students being selected on a nationally advertised competitive basis during the summer intersession period to perform research at Air Force Research Laboratory (AFRL) Technical Directorates and Air Force Air Logistics Centers (ALC). AFOSR also offers its research associates (faculty only) an opportunity, under the Summer Research Extension Program (SREP), to continue their AFOSR-sponsored research at their home institutions through the award of research grants. This volume consists of a listing of the participants for the SREP and the technical report from each participant working at the AF Rome Laboratory.

14. SUBJECT TERMS Air Force Research, Air Force, Engineering, Laboratories, Reports, Summer, Universities, Faculty, Graduate Student, High School Student			**15. NUMBER OF PAGES**
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements.***

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available
(e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C - Contract	**PR** - Project
G - Grant	**TA** - Task
PE - Program Element	**WU** - Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with....; Trans. of....; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD -	See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE -	See authorities.
NASA -	See Handbook NHB 2200.2.
NTIS -	Leave blank.

Block 12b. Distribution Code.

DOD -	Leave blank.
DOE -	Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
	Leave blank.
NASA -	Leave blank.
NTIS -	

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self- explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

1999 SREP Final Technical Report Table of Contents

Armstrong Research Laboratory

Volume 1

	Principle Investigator	Report Title University/Institution	Laboratory & Directorate
1	Dr. Kenneth Graetz	Conflict Resolution in Distributed Meetings:Using Collaboration Technology to Support Battle Staff University of Dayton	AFRL/HEN
2	Dr. Nadini Kannan	Altitude Decompression Sickness: Modeling and Prediction University of Texas at San Antonio	AFRL/HEPR
3	Ms. Vanessa Le	A Study on Stress-Induced Alterations In Blood-BrainBarrier Permeability to Pyridostigmine University of Texas at Austin	AFRL/HEDB
4	Dr. Ramaswamy Ramesh	Modeling and Analysis of DMT Systems: Training Effectiveness, Costs, Resource Management and Acquisition Strategies Research Foundation of SUNY	AFRL/HEA

1999 SREP Final Technical Report Table of Contents

Phillips Research Laboratory

Volume 2

	Principle Investigator	Report Title University/Institution	Laboratory & Directorate
1	Dr. Gurnam Gill	Adaptive signal Processing and its Applications in Space Space based Radar Naval Postgraduate School	AFRL/VSSS
2	Dr. Robert Hinde	Dopant-Induced Infrafed Activity in Solid Hydrogen An AB Initio and Quantum Monte Carlo Study University of Tennessee	AFRL/PRSP
3	Dr. Brian Jeffs	Algebraic Methods for Improved Blind Restoration of Adaptive Optics Images of Space Objects Brigham Young University	AFRL/DEHP
4	Dr. Donald Leo	Self-Sensing acoustic Sources For Interior Noise Control in Payload Fairings University of Toledo	AFRL/VSDV
5	Dr. Arfin Lodhi	Investigation into Time-Dependent Power Loses from AMTEC Components Texas Tech University	AFRL/VSDV
6	Dr. John McHugh	Atmospheric Gravity Waves Near the Tropopause University of New Mexico	AFRL/ VSBC
7	Dr. Stanly Steinberg	Lie-Algebraic representations of Product Integrals of Variable Matrices University of New Mexico	AFRL/DEOB
8	Mr. Kenneth Stephens II	Simulation of a Magnetized Target Fusion Concept Using MACH 2 University of North Texas	AFRL/VSDV

1999 SREP Final Technical Report Table of Contents

Rome Research Laboratory

Volume 3

Principle Investigator		Report Title University/Institution	Laboratory & Directorate
1	Dr. Ercument Arvas	Realization of Low Noise MMIC Amplifier as a Microwave –to Optics Link for Radar Syracuse University	AFRL/SNDR
2	Dr. Kaliappan Gopalan	Detection of Acoustic Correlates of Stress from Modulation Characteristics Purdue Research Foundation	AFRL/IFEC
3	Dr. Donald Hung	An Investigation on Accelerating the Ray-Tracing Computations Washington State University	AFRL/IFSA
4	Dr. Adam Lutoborski	Transform Methods for Watermarking Digital Images Syracuse University	AFRL/IFEC
5	Dr. Brajendra Panda	Implementation of Petri Nets Based Multi-source Attack Detection Model University of North Dakota	AFRL/IFGB
6	Dr. Jerry Potter	Algorithms for Data Intensive Knowledge Discovery Kent State University	AFRL/IFGA
7	Dr. Shambhu Upadhyaya	A Distributed concurrent Intrusion Detection and Recovery Scheme based on Assertions SUNY Buffalo	AFRL/IFGA

1999 SREP Final Technical Report Table of Contents

Wright Research Laboratory

Volume 4

Principle Investigator		Report Title University/Institution	Laboratory & Directorate
1	Dr. Farid Ahmed	Image Quality Assessment for ART Applications Using Multiresolional Information Metrics Pennsylvania State University, Erie	AFRL/SNAT
2	Dr. Gregory Buck	Acoustic Disturbance Source Modeling and Development for Hypersonic Receptivity South Dakota School of Mines	AFRL/VAAA
3	Dr. Kevin Belfield	Synthesis of New Two-Photon Absorbing Dyes, Monomers and Polymers University of Central Florida	AFRL/ML
4	Dr. Patrick Gilcrease	Biocatalysis of Biphenyl and Diphenylacetylene in an Aqueous Organic Biphasic Reaction System University of Wyoming	AFRL/MLQ
5	Dr. Jeffrey Johnson	Incorporating Fixed, Adaptive, and Learning Controllers to the Flight Control University of Toledo	AFRL/VACC
6	Dr. Vikram Kapila	Dynamics and control of Spacecraft Formation Flying Polytechnic Institute of New York	AFRL/VACC
7	Dr. Kenneth Kihm	Micro-Scale Flow Field Measurement of the Thin Meniscus of Capillary-Driven Heat Exchanger Devices Using MFW Texas A & M University	AFRL/VAVE
8	Dr. Rongxing Li	Uncertainty Modeling of Target Locations From Multiplatform and Multisensor Data Ohio State	AFRL/SNAR
9	Dr. Chun-Shin Lin	Sensor Fusion w/Passive Millimeter Wave & Laser Radar for Target Detection University of Missouri –	AFRL/MNGS
10	Dr. Chaoqun Liu	Boundary Conditions in Curvilinear Coordinates for Direct Numerical Simulation of Turbulent Flow Louisiana Tech University	AFRL/VAAA

1999 SREP Final Technical Report Table of Contents

Wright Research Laboratory

Volume 4, continued

	Principle Investigator	Report Title University/Institution	Laboratory & Directorate
11	Dr. Carl Mungan	Infrared Spectropolarimetric Directional Reflectance and Emissivity of Mental Surfaces University of Forida	AFRL/MNGS
12	Dr. Amod Ogale	Structural Changes in Mesophase Pitch-Based Carbon Fibers: In SITU & ES SITU Measurements Clemson University	AFRL/MLBC
13	Mr. Ramana Pidaparti	Benchmarking Aerodynamic Panel Methods for Flight Loads in Multidisciplinary Optimization Indiana University	AFRL/VAS
14	Dr. Stephen Saddow	Silicon Carbide Implant Activation & Surface preparation Investigation Mississippi State University	AFRL/PRPE
15	Dr. Paavo Sepri	Computational Study of Unsteady Flow Interactions Between Turbine Blades, Cylinder Wakes, and coolant Injection Florida Institute of Technology	AFRL/PRTT
16	Dr. Hongchi Shi	Developing an efficient Algorithm for Routing Processors of the VGI Parallel Computer for Signal Processing Applications University of Missouri – Columbia	AFRL/MNGI
17	Dr. Mehrdad Soumekh	Signal and Image Processing for FOPEN/GPEN SAR SUNY Buffalo	AFRL/SNRT
18	Mr. Craig Riviello	In-Situ Synthesis of Discontinuously Reinforced Titanium alloy CompoLaboratorys Via Blended Elemental Powder Metallurgy Processing Wright State University	AFRL/MLLM

1999 SUMMER RESEARCH EXTENSION PROGRAM (SREP) MANAGEMENT REPORT

1.0 BACKGROUND

Under the provisions of Air Force Office of Scientific Research (AFOSR) contract F49620-90-C-0076, September 1990, Research & Development Laboratories (RDL), an 8(a) contractor in Culver City, CA, manages AFOSR's Summer Research Program. This report is issued in partial fulfillment of that contract (CLIN 0003AC).

The Summer Research Extension Program (SREP) is one of four programs AFOSR manages under the Summer Research Program. The Summer Faculty Research Program (SFRP) and the Graduate Student Research Program (GSRP) place college-level research associates in Air Force research laboratories around the United States for 8 to 12 weeks of research with Air Force scientists. The High School Apprenticeship Program (HSAP) is the fourth element of the Summer Research Program, allowing promising mathematics and science students to spend two months of their summer vacations working at Air Force laboratories within commuting distance from their homes.

SFRP associates and exceptional GSRP associates are encouraged, at the end of their summer tours, to write proposals to extend their summer research during the following calendar year at their home institutions. AFOSR provides funds adequate to pay for SREP subcontracts. In addition, AFOSR has traditionally provided further funding, when available, to pay for additional SREP proposals, including those submitted by associates from Historically Black Colleges and Universities (HBCUs) and Minority Institutions (MIs). Finally, laboratories may transfer internal funds to AFOSR to fund additional SREPs. Ultimately the laboratories inform RDL of their SREP choices, RDL gets AFOSR approval, and RDL forwards a subcontract to the institution where the SREP associate is employed. The subcontract (see Appendix 1 for a sample) cites the SREP associate as the principal investigator and requires submission of a report at the end of the subcontract period.

Institutions are encouraged to share costs of the SREP research, and many do so. The most common cost-sharing arrangement is reduction in the overhead, fringes, or administrative charges institutions would normally add on to the principal investigator's or research associate's labor. Some institutions also provide other support (e.g., computer run time, administrative assistance, facilities and equipment or research assistants) at reduced or no cost.

When RDL receives the signed subcontract, we fund the effort initially by providing 90% of the subcontract amount to the institution (normally $18,000 for a $20,000 SREP). When we receive the end-of-research report, we evaluate it administratively and send a copy to the laboratory for a technical evaluation. When the laboratory notifies us the SREP report is acceptable, we release the remaining funds to the institution.

2.0 THE 1999 SREP PROGRAM

SELECTION DATA: A total of 381 faculty members (SFRP Associates) and 130 graduate students (GSRP associates) applied to participate in the 1998 Summer Research Program. From these applicants 85 SFRPs and 40 GSRPs were selected. The education level of those selected was as follows:

1998 SRP Associates, by Degree			
SFRP		GSRP	
PHD	MS	MS	BS
83	1	3	19

Of the participants in the 1998 Summer Research Program 65 percent of SFRPs and 20 percent of GSRPs submitted proposals for the SREP. Fifty-four proposals from SFRPs and eleven from GSRPs were selected for funding, which equates to a selection rate of 65% of the SFRP proposals and of 20% for GSRP proposals.

1999 SREP: Proposals Submitted vs. Proposals Selected			
	Summer 1998 Participants	Submitted SREP Proposals	SREPs Funded
SFRP	85	54	34
GSRP	40	11	2
TOTAL	125	65	36

The funding was provided as follows:

Contractual slots funded by AFOSR	36
Laboratory funded	0
Total	36

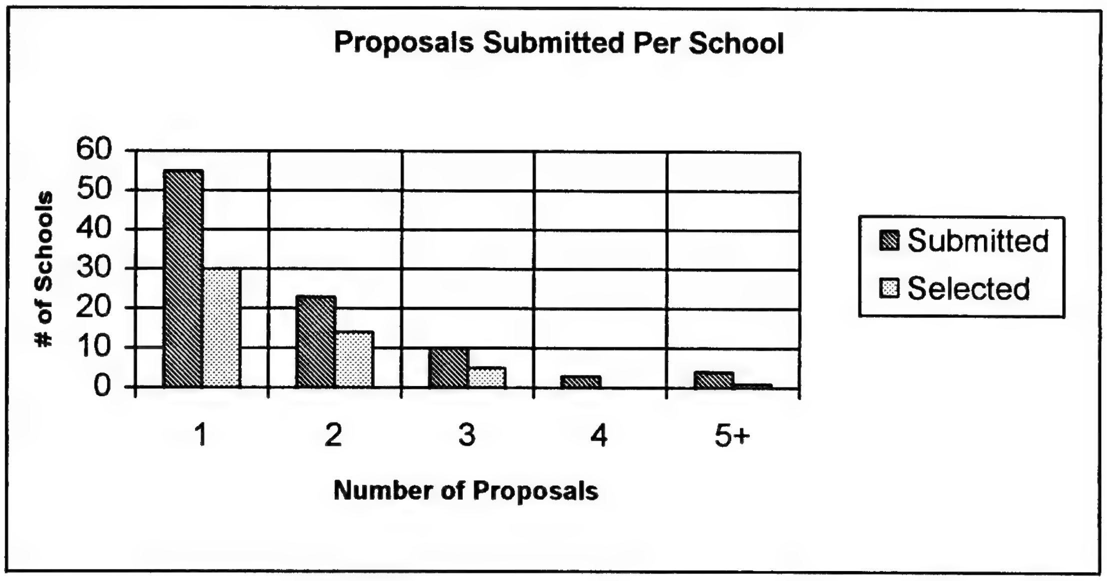
Four HBCU/MI associates from the 1998 summer program submitted SREP proposals; four were selected (none were lab-funded; all were funded by additional AFOSR funds).

Proposals Submitted and Selected, by Laboratory		
	Applied	Selected
Armstrong Research Site	15	3
Air Logistic Centers	1	0
Arnold Engineering Development Center	1	0
Phillips Research Site	10	8
Rome Research Site	12	7
Wilford Hall Medical Center	1	0
Wright Research Site	25	18
TOTAL	65	36

The 212 1998 Summer Research Program participants represented 60 institutions.

Institutions Represented on the 1998 SRP and 1999 SREP		
Number of schools Represented in the Summer 98 Program	Number of schools represented in submitted proposals	Number of schools represented in Funded Proposals
60	36	29

Thirty schools had more than one participant submitting proposals.



The selection rate for the 60 schools submitting 1 proposal (68%) was better than those submitting 2 proposals (61%), 3 proposals (50%), 4 proposals (0%) or 5+ proposals (25%). The 4 schools that submitted 5+ proposals accounted for 30 (15%) of the 65 proposals submitted.

Of the 65 proposals submitted, 35 offered institution cost sharing. Of the funded proposals which offered cost sharing, the minimum cost share was $1274.00, the maximum was $38,000.00 with an average cost share of $12,307.86.

Proposals and Institution Cost Sharing

	Proposals Submitted	Proposals Funded
With cost sharing	35	30
Without cost sharing	30	6
Total	65	36

The SREP participants were residents of 31 different states. Number of states represented at each laboratory were:

States Represented, by Proposals Submitted/Selected per Laboratory

	Proposals Submitted	Proposals Funded
Armstrong Research Laboratory	15	3
Air Logistic Centers	1	0
Arnold Engineering Development Center	1	0
Phillips Research Laboratory	10	8
Rome Research Laboratory	12	7
Wilford Hall Medical Center	1	0
Wright Research Laboratory	25	18

Six of the 1999 SREP Principal Investigators also participated in the 1998 SREP.

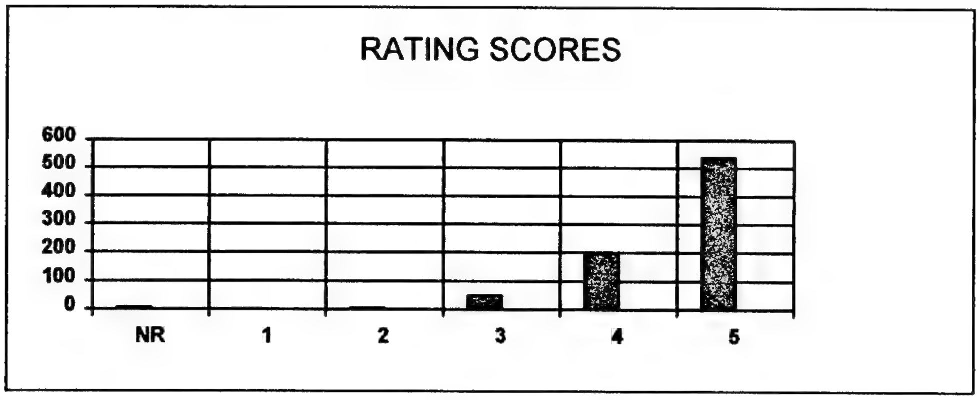
ADMINISTRATIVE EVALUATION: The administrative quality of the SREP associates' final reports was satisfactory. Most complied with the formatting and other instructions provided to them by RDL. Thirty-six final reports have been received and are included in this report. The subcontracts were funded by $897,309.00 of Air Force money. Institution cost sharing totaled $356,928.00.

TECHNICAL EVALUATION: The form used for the technical evaluation is provided as Appendix 2. Thirty-two evaluation reports were received. Participants by laboratory versus evaluations submitted is shown below:

	Participants	Evaluations	Percent
Armstrong Laboratory	3	2	95.2
Phillips Laboratory	8	8	100
Rome Laboratory	7	7	100
Wright Laboratory	18	15	91.9
Total	36	32	95.0

Notes:
1: Research on four of the final reports was incomplete as of press time so there aren't any technical evaluations on them to process, yet. Percent complete is based upon 20/21=95.2%

PROGRAM EVALUATION: Each laboratory focal point evaluated ten areas (see Appendix 2) with a rating from one (lowest) to five (highest). The distribution of ratings was as follows:

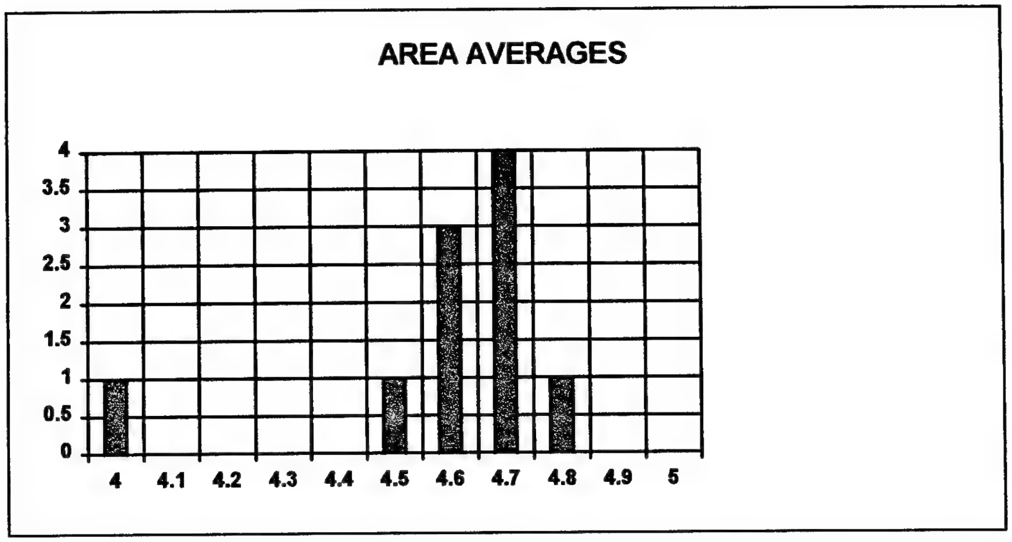


Rating	Not Rated	1	2	3	4	5
# Responses	7	1	7	62 (6%)	226 (25%)	617 (67%)

The 8 low ratings (one 1 and seven 2's) were for question 5 (one 2) "The USAF should continue to pursue the research in this SREP report" and question 10 (one 1 and six 2's) "The one-year period for complete SREP research is about right", in addition over 30% of the threes (20 of 62) were for question ten. The average rating by question was:

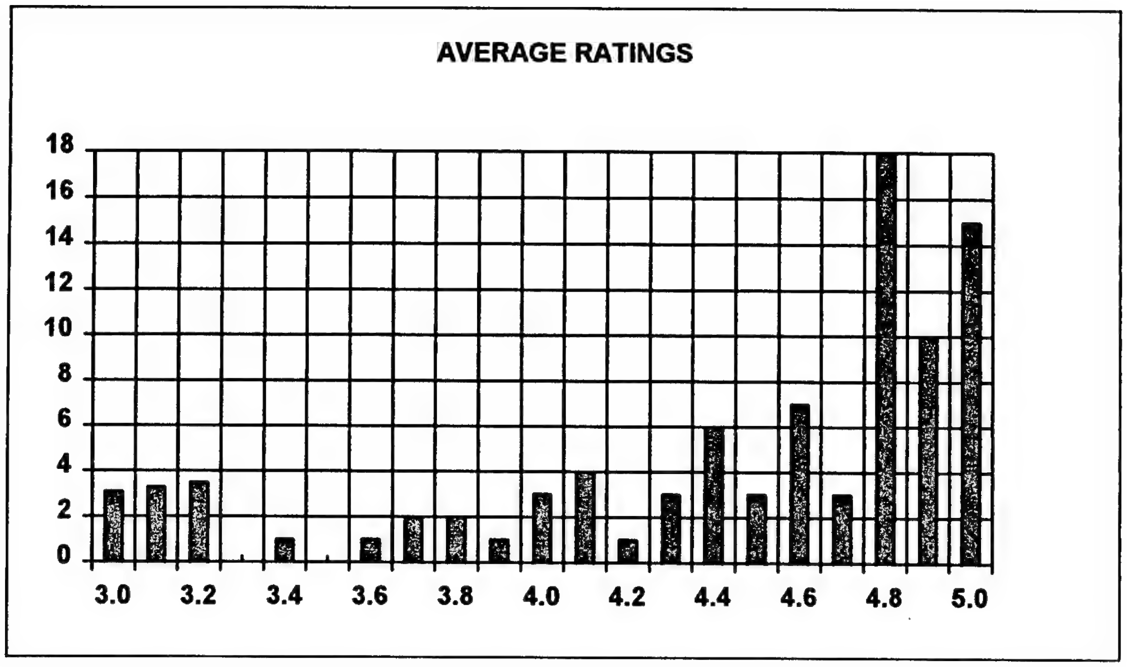
Question	1	2	3	4	5	6	7	8	9	10
Average	4.6	4.6	4.7	4.7	4.6	4.7	4.8	4.5	4.6	4.0

The distribution of the averages was:



Area 10 "the one-year period for complete SREP research is about right" had the lowest average rating (4.1). The overall average across all factors was 4.6 with a small sample standard deviation of 0.2. The average rating for area 10 (4.1) is approximately three sigma lower than the overall average (4.6) indicating that a significant number of the evaluators feel that a period of other than one year should be available for complete SREP research.

The average ratings ranged from 3.4 to 5.0. The overall average for those reports that were evaluated was 4.6. Since the distribution of the ratings is not a normal distribution the average of 4.6 is misleading. In fact over half of the reports received an average rating of 4.8 or higher. The distribution of the average report ratings is as shown:



It is clear from the high ratings that the laboratories place a high value on AFOSR's Summer Research Extension Programs.

3.0 SUBCONTRACTS SUMMARY

Table 1 provides a summary of the SREP subcontracts. The individual reports are published in volumes as shown:

Laboratory	Volume
Armstrong Research Laboratory	1
Phillips Research Laboratory	2
Rome Research Laboratory	3
Wright Research Laboratory	4

SREP SUB-CONTRACT DATA

Report Author Author's University	Author's Degree	Sponsoring Lab	Performance Period		Contract Amount	Univ. Cost Share
Graetz , Kenneth Department of Psychology University of Dayton, Dayton, OH	PhD 99-0823	AL/CF Conflict resolution in distributed Meetingl: Using Collaboration Technology to	01/01/99	12/31/99	$24983.00	$0.00
Kannan , Nandini Statistics Univ of Texas at San Antonio, San Antonio, TX	PhD 99-0824	AL/CF Altitude decompression Sickness: Modeling and Prediction	01/01/98	12/31/98	$22492.00	$4478.00
Ramesh , Ramaswamy Magement Science/Systems Research Foundation of SUNY, Buffalo, NY	PhD 99-0822	AL/CF Modeling and Analsis of DMT Systems: Training Effectiveness, Costs, Resource Ma	01/01/99	12/31/99	$24979.00	$0.00
Le , Vanessa Biochemistry Univ of Texas at Austin, Austin, TX	BS 99-0825	AL/CF a Study on Stress-Induced Alterations In Blood-Brain Barrier Permeability to Pyr	01/01/98	12/31/98	$25000.00	$0.00
Gill , Gurnam EE Naval Postgraduate School, Monterey, CA	PhD 99-0834	PL/VT Adaptive signal Processing and its Applications in Space based Radar	01/01/99	12/31/99	$25000.00	$0.00
Hinde , Robert Physical Chemistry Univ of Tennessee, Knoxville, TN	PhD 99-0831	WL/PO Dopant-Induced Infrafed Activity in Solid Hydrogen An AB Initio and Quantum Mont	01/01/98	12/31/98	$25000.00	$3976.00
Jeffs , Brian Electrical Engineering Brigham Young University, Provo, UT	PhD 99-0828	PL/LI Algebraic Methods for Improved blimd Restoration of Adaptive Optics Images of Sp	01/01/98	12/31/98	$25000.00	$19177.00
Leo , Donald Mechanical & Aerospace Virginia Tech, Blacksburg, VA	PhD 99-0835	PL/VT Self-Sensing Acoustic Sources For Interior Noise Control in Payload Fairings	01/01/99	12/31/99	$24999.00	$7416.00
Lodhi , M. Arfin Nuclear Physics Texas Tech University, Lubbock, TX	PhD 99-0832	PL/VT Investigatioin into Time-Dependent Power Losses from AMTEC Components	01/01/99	12/31/99	$25000.00	$0.00
McHugh , John Applied Mechanics University of New Hampshire, Durham, NH	PhD 99-0833	PL/VT Atmospheric Gravity Waves Near The Tropopause	01/01/99	12/13/99	$25000.00	$7000.00
Steinberg , Stanly Mathematics University of New Mexico, Albuquerque, NM	PhD 99-0829	PL/LI Lie-Algebraic Representations of Product Integrals of Variable Matrices	01/01/99	12/31/99	$25000.00	$0.00
Stephens II , Kenneth University of North Texas, Denton, TX	MA 99-0830	PL/LI Simulation of a Magnetized Target Fusion Concept Using MACH2	01/01/99	12/31/99	$25000.00	$16764.00
Arvas , Ercument Electrical Engineering Syracuse University, Syracuse, NY	PhD 99-0838	WL/AA Realization of Low Noise MMIC Amplifier as a Microwave-to-Optics Link for Radar	01/01/99	12/31/99	$25000.00	$13000.00
Gopalan , Kaliappan Electrical Engineering Purdue Research Foundation, West Lafayette, IN	PhD 99-0814	RL/IR Detection of Acoustic Correlates of Stress from Modulation Characteristics	01/01/99	12/31/99	$25000.00	$38168.00

SREP SUB-CONTRACT DATA

Report Author Author's University	Author's Degree	Sponsoring Lab	Performance Period	Contract Amount	Univ. Cost Share
Hung , Donald Electrical Engineering Washington State University, Richland, WA	PhD 99-0812	RL/IR	01/01/99 12/31/99	$25000.00	$23008.00
		An Investigation on Accelerating the Ray-Tracing Compputations			
Lutoborski , Adam Applied Mathematics Syracuse University, Syracuse, NY	PhD 99-0811	RL/IR	01/01/99 12/31/99	$25000.00	$1274.00
		Transform Methods for Watermarking Digital Images			
Panda , Brajendra Computer Science University of North Dakota, Grand Forks, ND	PhD 99-0810	RL/IR	01/01/98 12/31/98	$24942.00	$2600.00
		Implementation of Petri Nets Based Multi-source Attack Detection Model			
Potter , Jerry Computer Science Kent State University, Kent, OH	PhD 99-0809	RL/IR	01/01/99 12/31/99	$25000.00	$52767.00
		Algorithms for Data Intensive Knowledge Discovery			
Upadhyaya , Shambhu Elec & Comp Engineering SUNY Buffalo, Buffalo, NY	PhD 99-0813	RL/IR	01/01/98 12/31/98	$25000.00	$6430.00
		a Distrubuted concurrent Intrusion Detection And recovery Scheme based on Assert			
Ahmed , Farid Electrical engineering Penn State Uni-Erie, Erie, PA	PhD 99-0806	WL/AA	10/10/98 12/31/98	$25000.00	$2396.00
		Image Quality Assessment for ATR Applications Using Multiresolutional Informatio			
Belfield , Kevin Chemistry University of Central Florida, Orlando, FL	PhD 99-0816	WL/ML	01/01/99 12/31/99	$25000.00	$5765.00
		Synthesis of New Two-Photon Absorbing Dyes, Monomers and Polymers			
Buck , Gregory Mechanical Engineering S Dakota School of Mines/Tech, Rapid City, SD	PhD 99-0818	WL/FI	01/01/99 12/31/99	$25000.00	$7639.00
		Acoustic Disturbance Source Modeling and Development for Hypersonic Receptivity			
Gilcrease , Patrick Chemical Engineering University of Wyoming, Laramie, WY	PhD 99-0815	WL/ML	01/01/99 12/31/99	$25000.00	$28010.00
		Biocatalysis of Biphenyl and Diphenylacetylene in an Aqueous-Organic Biphasic Re			
Johnson , Jeffrey Electrical Engineering and University of Toledo, Toledo, OH	PhD 99-0823	WL/FI	01/01/99 12/31/99	$25000.00	$10075.00
		Incorporating Fixed, Adaptive, & Learning Controllers to the Flight Control			
Kapila , Vikram Aerospace engineering Polytechnic Inst of New York, Brooklyn, NY	PhD 99-0820	WL/FI	01/01/99 12/31/99	$25000.00	$17448.00
		Dynamics and Control of Spacecraft Formation Flying			
Kihm , Kenneth Mechanical Engineering Texas Engineering Experiment Station, College	PhD 99-0821	WL/FI	01/01/99 12/31/99	$25000.00	$10310.00
		Micro-Scale Flow Field Measurement of the Thin Meniscus of Capillary-Driven Heat			
Li , Rongxing Photogrammertry & Remote Sensing Ohio State University, Columbus, OH	PhD 99-0831	WL/AA	01/01/98 12/31/98	$25000.00	$13183.00
		Uncertainty Modeling of Target Locations From Multiplatform and Multisensor Data			
Lin , Chun-Shin Electrical Engineering Univ of Missouri - Columbia, Columbia, MO	PhD 99-0826	WL/MN	01/01/99 12/31/99	$25000.00	$1991.00
		Sensor Fusion w/Passive Millimeter Wave & Laser Radar for Target Detection			
Liu , Chaoqun Applied Mathematics Louisiana Tech University, Ruston, LA	PhD 99-0819	WL/FI	01/01/99 12/31/99	$25000.00	$12521.00
		Boundary Conditions in Curvilinear Coordinates for Direct Numerical Simulation			

SREP SUB-CONTRACT DATA

Report Author Author's University	Author's Degree	Sponsoring Lab	Performance Period		Contract Amount	Univ. Cost Share
Mungan , Carl	PhD	WL/MN	01/01/99	12/31/99	$24914.00	$3276.00
Dept of Physics	99-0824	infrared Spectropolarimetric Directonal				
University of Florida. Pensacola, FL		Reflectance and Emissivity of Mental Sur				
Ogale , Amod	PhD	WL/ML	01/01/99	12/31/99	$25000.00	$9000.00
Chemical Engineering	99-0817	Structural Changes in Mesophase Pitch-Based				
Clemson University, Clemson, SC		Carbon Fibers:In SITU &ES SITU Measu				
Pidaparti , Ramana	PhD	WL/FI	01/01/99	12/31/99	$25000.00	$10582.00
Aeronautics & Astronautics	99-0822	Benchmarking Aerodynamic Panel Methods for				
Indiana U-Purdue at Indianap, Indianapolis, IN		Flight Loads in Multidisciplinary Opt				
Saddow , Stephen	PhD	WL/PO	01/01/98	12/31/98	$25000.00	$0.00
Electrical Engineering	99-0827	Silicon Carbide Implant Activation & Surface				
Mississippi State University, Mississippi State,		preparation Investigation				
Sepri , Paavo	PhD	WL/PO	01/01/99	12/31/99	$25000.00	$6519.00
Engineering Science	99-0836	Computational Study of Unsteady Flow				
Florida Inst of Technology, Melbourne, FL		Interactions Between Turbine Blades, Cylind				
Shi , Hongchi		WL/MN	01/01/99	12/31/99	$25000.00	$15851.00
Computer Engineering	99-0825	Developing an efficient Algorithm for Routing				
Univ of Missouri - Columbia, Columbia, MO		Processors of the VGI Parallel Com				
Soumekh , Mehrdad	PhD	WL/AA	01/01/99	12/30/99	$25000.00	$0.00
Elec/Computer Engineering	99-0807	Signal and Image Processing for FOPEN/GPEN SAR				
SUNY Buffalo, Amherst, NY						
Riviello , Craig	BS	WL/ML	01/01/99	01/01/99	$25000.00	$6304.00
Mechanical	99-0837	In-Situ Synthesis of Discontinuously Reinforced				
Wright State University, Dayton, OH		Titanium alloy Composites Via Bl				

APPENDIX 1:

SAMPLE SREP SUBCONTRACT

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
1999 SUMMER RESEARCH EXTENSION PROGRAM
SUBCONTRACT 99-0806

BETWEEN

Research & Development Laboratories
5800 Uplander Way
Culver City,CA 90230-6608

AND

Penn State Erie, The Behrend College
Contracts and Grants Office
Erie, PA 16563

REFERENCE:	Summer Research Extension Program Proposal			98-0029
	Start Date:	01/01/99	End Date:	12/31/99
	Proposal Amount:	$25,000		
	Proposal Title:	Image Quality Assessment for ATR Applications Using Multiresolutional Informational Information Metrics		

PRINCIPAL INVESTIGATOR: Dr. Farid Ahmed
Penn State Erie, The Behrend College
Erie, PA 16563

(2) UNITED STATES AFOSR CONTRACT NUMBER: F49620-93-C-0063

(3) CATALOG OF FEDERAL DOMESTIC ASSISTANCE NUMBER (CFDA):12.800
PROJECT TITLE: AIR FORCE DEFENSE RESEARCH SOURCES PROGRAM

(4) ATTACHMENTS
1 REPORT OF INVENTIONS AND SUBCONTRACT
2 CONTRACT CLAUSES
3 FINAL REPORT INSTRUCTIONS

*****SIGN SREP SUBCONTRACT AND RETURN TO RDL*****

1. BACKGROUND: Research & Development Laboratories (RDL) is under contract (F49620-93-C-0063) to the United States Air Force to administer the Summer Research Program (SRP), sponsored by the Air Force Office of Scientific Research (AFOSR), Bolling Air Force Base, D.C. Under the SRP, a selected number of college faculty members and graduate students spend part of the summer conducting research in Air Force laboratories. After completion of the summer tour participants may submit, through their home institutions. proposals for follow-on research. The follow-on research is known as the Summer Research Extension Program (SREP). Approximately 61 SREP proposals annually will be selected by the Air Force for funding of up to $25,000; shared funding by the academic institution is encouraged. SREP efforts selected for funding are administered by RDL through subcontracts with the institutions. This subcontract represents an agreement between RDL and the institution herein designated in Section 5 below.

2. RDL PAYMENTS: RDL will provide the following payments to SREP institutions:

   - 80 percent of the negotiated SREP dollar amount at the start of the SREP research period.
   - The remainder of the funds within 30 days after receipt at RDL of the acceptable written final report for the SREP research.

3. INSTITUTION'S RESPONSIBILITIES: As a subcontractor to RDL, the institution designated on the title page will:

a. Assure that the research performed and the resources utilized adhere to those defined in the SREP proposal.

b. Provide the level and amounts of institutional support specified in the SREP proposal..

c. Notify RDL as soon as possible, but not later than 30 days, of any changes in 3a or 3b above, or any change to the assignment or amount of participation of the Principal Investigator designated on the title page.

d. Assure that the research is completed and the final report is delivered to RDL not later than twelve months from the effective date of this subcontract, but no later than December 31, 1998. The effective date of the subcontract is one week after the date that the institution's contracting representative signs this subcontract, but no later than January 15, 1998.

e. Assure that the final report is submitted in accordance with Attachment 3.

f. Agree that any release of information relating to this subcontract (news releases, articles, manuscripts, brochures, advertisements, still and motion pictures, speeches, trade associations meetings, symposia, etc.) will include a statement that the project or effort depicted was or is sponsored by: Air Force Office of Scientific Research, Bolling AFB, D.C.

g. Notify RDL of inventions or patents claimed as the result of this research as specified in Attachment 1.

h. RDL is required by the prime contract to flow down patent rights and technical data requirements to this subcontract. Attachment 2 to this subcontract contains a list of contract clauses incorporated by reference in the prime contract.

4. All notices to RDL shall be addressed to:

RDL AFOSR Program Office
5800 Uplander Way
Culver City, CA 90230-6609

5. By their signatures below, the parties agree to provisions of this subcontract.

______________________	______________________
Abe Sopher RDL Contracts Manager	Signature of Institution Contracting Official
	______________________
	Typed/Printed Name
______________________	______________________
Date	Title
	______________________
	Institution
	______________________
	Date/Phone

ATTACHMENT 2
CONTRACT CLAUSES

This contract incorporates by reference the following clauses of the Federal Acquisition Regulations (FAR), with the same force and effect as if they were given in full text. Upon request, the Contracting Officer or RDL will make their full text available (FAR 52.252-2).

FAR CLAUSES	TITLE AND DATE
52.202-1	DEFINITIONS
52.203-3	GRATUITIES
52.203-5	COVENANT AGAINST CONTINGENT FEES
52.203-6	RESTRICTIONS ON SUBCONTRACTOR SALES TO THE GOVERNMENT
52.203-7	ANTI-KICKBACK PROCEDURES
52.203-8	CANCELLATION, RECISSION, AND RECOVERY OF FUNDS FOR ILLEGAL OR IMPROPER ACTIVITY
52.203-10	PRICE OR FEE ADJUSTMENT FOR ILLEGAL OR IMPROPER ACTIVITY
52.203-12	LIMITATION ON PAYMENTS TO INFLUENCE CERTAIN FEDERAL TRANSACTIONS
52.204-2	SECURITY REQUIREMENTS
52.209-6	PROTECTING THE GOVERNMENT'S INTEREST WHEN SUBCONTRACTING WITH CONTRACTORS DEBARRED, SUSPENDED, OR PROPOSED FOR DEBARMENT
52.212-8	DEFENSE PRIORITY AND ALLOCATION REQUIREMENTS
52.215-2	AUDIT AND RECORDS - NEGOTIATION
52.215-10	PRICE REDUCTION FOR DEFECTIVE COST OR PRICING DATA

52.215-12	SUBCONTRACTOR COST OR PRICING DATA
52.215-14	INTEGRITY OF UNIT PRICES
52.215-8	ORDER OF PRECEDENCE
52.215.18	REVERSION OR ADJUSTMENT OF PLANS FOR POSTRETIREMENT BENEFITS OTHER THAN PENSIONS
52.222-3	CONVICT LABOR
52.222-26	EQUAL OPPORTUNITY
52.222-35	AFFIRMATIVE ACTION FOR SPECIAL DISABLED AND VIETNAM ERA VETERANS
52.222-36	AFFIRMATIVE ACTION FOR HANDICAPPED WORKERS
52.222-37	EMPLOYMENT REPORTS ON SPECIAL DISABLED VETERAN AND VETERANS OF THE VIETNAM ERA
52.223-2	CLEAN AIR AND WATER
52.223-6	DRUG-FREE WORKPLACE
52.224-1	PRIVACY ACT NOTIFICATION
52.224-2	PRIVACY ACT
52.225-13	RESTRICTIONS ON CONTRACTING WITH SANCTIONED PERSONS
52.227-1	ALT. I - AUTHORIZATION AND CONSENT
52.227-2	NOTICE AND ASSISTANCE REGARDING PATIENT AND COPYRIGHT INFRINGEMENT

52.227-10	FILING OF PATENT APPLICATIONS - CLASSIFIED SUBJECT MATTER
52.227-11	PATENT RIGHTS - RETENTION BY THE CONTRACTOR (SHORT FORM)
52.228-7	INSURANCE - LIABILITY TO THIRD PERSONS
52.230-5	COST ACCOUNTING STANDARDS - EDUCATIONAL INSTRUCTIONS
52.232-23	ALT. I - ASSIGNMENT OF CLAIMS
52.233-1	DISPUTES
52.233-3	ALT. I - PROTEST AFTER AWARD
52.237-3	CONTINUITY OF SERVICES
52.246-25	LIMITATION OF LIABILITY - SERVICES
52.247-63	PREFERENCE FOR U.S. - FLAG AIR CARRIERS
52.249-5	TERMINATION FOR CONVENIENCE OF THE GOVERNMENT (EDUCATIONAL AND OTHER NONPROFIT INSTITUTIONS)
52.249-14	EXCUSABLE DELAYS
52.251-1	GOVERNMENT SUPPLY SOURCES

DOD FAR CLAUSES	DESCRIPTION
252.203-7001	SPECIAL PROHIBITION ON EMPLOYMENT
252.215-7000	PRICING ADJUSTMENTS
252.233-7004	DRUG FREE WORKPLACE (APPLIES TO SUBCONTRACTS WHERE THERE IS ACCESS TO CLASSIFIED INFORMATION)
252.225-7001	BUY AMERICAN ACT AND BALANCE OF PAYMENTS PROGRAM
252.225-7002	QUALIFYING COUNTRY SOURCES AS SUBCONTRACTS
252.227-7013	RIGHTS IN TECHNICAL DATA - NONCOMMERCIAL ITEMS
252.227-7030	TECHNICAL DATA - WITHOLDING PAYMENT
252.227-7037	VALIDATION OF RESTRICTIVE MARKINGS ON TECHNICAL DATA
252.231-7000	SUPPLEMENTAL COST PRINCIPLES
252.232-7006	REDUCTIONS OR SUSPENSION OF CONTRACT PAYMENTS UPON FINDING OF FRAUD

APPENDIX 2:

SAMPLE TECHNICAL EVALUATION FORM

SUMMER RESEARCH EXTENSION PROGRAM
TECHNICAL EVALUATION

SREP NO: 99-0828
PRINCIPAL INVESTIGATOR: Dr.Brian Jeffs
Brigham Young University

Circle the rating level number, 1 (low) through 5 (high), you feel best evaluate each statement and return the completed form by fax or mail to:

RDL
Attn: SREP Tech Evals
5800 Uplander Way
Culver City, CA 90230-6608

1. This SREP report has a high level of technical merit	1 2 3 4 5
2. The SREP program is important to accomplishing the lab's mission	1 2 3 4 5
3. This SREP report accomplished what the associate's proposal promised	1 2 3 4 5
4. This SREP report addresses area(s) important to USAF	1 2 3 4 5
5. The USAF should continue to pursue the research in this SREP report	1 2 3 4 5
6. The USAF should maintain research relationships with this SREP associate	1 2 3 4 5
7. The money spent on this SREP effort was well worth it.	1 2 3 4 5
8. This SREP report is well organized and well written	1 2 3 4 5
9. I'll be eager to be a focal point for summer and SREP associates in the future	1 2 3 4 5
10.The one-year period for complete SREP research is about right.	1 2 3 4 5

11. If you could change any one thing about the SREP program, what would you change:

12. What do you definitely NOT change about the SREP program?

PLEASE USE THE BACK FOR ANY OTHER COMMENTS

Laboratory: AFRL/IFEC
Lab Focal Point: Mr. Stanley Wendt
PHONE: (315) 330-7244

REALIZATION OF A LOW-NOISE MMIC AMPLIFIER AS A MICROWAVE-TO-OPTICS LINK FOR RADAR APPLICATIONS

Ercument Arvas
Professor
Department of Electrical Engineering and Computer Science

121 Link Hall
Syracuse University
Syracuse, NY 13244

Final Report for:
Summer Research Extension Program

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Rome Laboratory

December 1999

REALIZATION OF A LOW-NOISE MMIC AMPLIFIER AS A MICROWAVE-TO-OPTICS LINK FOR RADAR APPLICATIONS

Ercument Arvas
Professor
Department of Electrical Engineering and Computer Science
Syracuse University

Abstract

A MMIC distributed amplifier with 12-dB small signal gain in 3.1-3.5 GHz band was designed, simulated and built. A very flat gain in the desired bandwidth is the primary design specification of the amplifier since it is required to feed an optical system. Distributed amplifier topology is preferred since it is difficult to design very flat-gain amplifiers using traditional single-stage or multiple stage approach. In addition to flat gain requirement, good input, and output return losses with a noise figure of less than five dB are to be realized. The GaAs monolithic circuit technology was used and the foundry process TQTRX of TriQuint Semiconductor was chosen. With this process, it is possible to have designs with good operating performances up to 10 GHz. The amplifier has 50 Ω input and output terminations.

REALIZATION OF A LOW-NOISE MMIC AMPLIFIER AS A MICROWAVE-TO-OPTICS LINK FOR RADAR APPLICATIONS

Ercument Arvas

1. Introduction

The use of Microwave Monolithic Integrated Circuits (MMIC) in space hardware is becoming increasingly common. This is due to several advantages: reduction in mass and size, reliability improvement, repeatability of performance and reduction of parasitics.

GaAs technology is used in many RF systems. Integrated GaAs technology is involved in cellular phones, cable TV, and satellite transceiver and in many other wireless data applications. Telecommunications and computing fields use this technology in fiber optic receivers and high-speed data paths. Factors like linearity, noise consideration, power efficiency, ease of design and system cost makes GaAs natural for wireless applications. Also low voltage RF systems favor GaAs technology. The frequency span of the GaAs technology as compared to other technologies is illustrated in Fig.1.1.

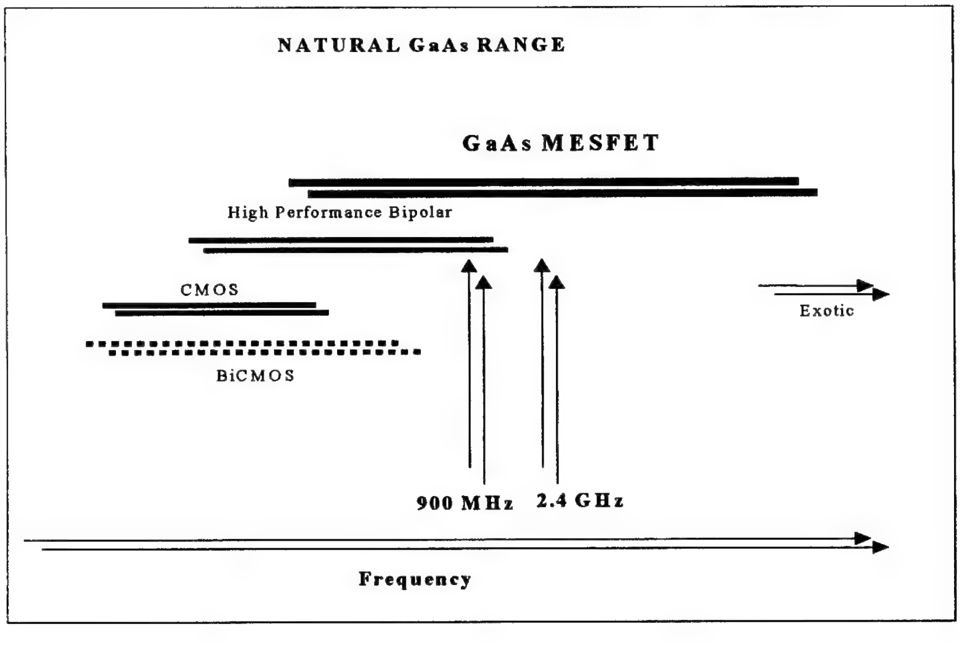


Fig.1.1. Natural GaAs Range [5].

The low frequency nature of silicon makes it useless for applications running in the GHz ranges. GaAs plays an important role in wireless systems when used with silicon. In today's wireless systems RF signal conversion is done in GaAs following the signal processing in Silicon. The main building blocks in this kind of systems are frequency up or down converters, frequency synthesizers, power amplifiers, low noise amplifiers, and transmit/receive switches.

GaAs is similar to RF Si Bipolar in terms of cost per area of semiconductor. Same packages and suppliers in Si technology can be used in GaAs also. GaAs RF integration reduces space and parts count. There are only 13 layers in GaAs processing versus 21 layers in Si for the same function. Material costs are dropping rapidly. Many technologies require power efficiency and good linearity/distortion performance. This makes GaAs the most demanded technology [5].

The design of MMICs must be performed with particular care and attention as circuit tuning is practically impossible after manufacture, and each design fabrication iteration is very expensive. In addition, manufacturing process is very long and sometimes it might take one year if one needs to do a few iterations to make a circuit function properly. Designer should be very careful and check all the design specifications before the real manufacturing process is started. TQTRX process will be introduced shortly in the following section. Then the method of MMIC Distributed Amplifier Design will be summarized in Section 3. Finally, the design steps and simulation results are given in Section 4.

2. TQTRX Process Technology Overview

TQTRX is one of the best process for high performance transceivers. This may not be the best process option for our purpose but it goes with the Prototype Chip Option (PCO) since the design will be a prototype. The other processes are HA2 (general purpose RF + power), TQHIP (highest power RFICs), QEDA2 (high-density mixed mode) [2].

2.1 Typical Applications

TQTRx is a high performance GaAs E/D MESFET based process for RF transceiver applications and mixed mode applications up to LSI complexity. The process features high performance enhancement and depletion mode MESFETs for small signal and a depletion mode power MESFET for power amplifier stages. Utilizing a unique thick plated gold interconnection and highly robust interlayer dielectric, the process is optimized for ease of automated assembly. This also enables the use of low cost injection molded plastic packaging without special handling or fixturing. TQTRx features very high value capacitors for increased density and reduced die size . It also features precision NiCr resistors.

Some essential electrical parameters are listed for reference only in Table 2.1.

Table 2.1. TQTRx Key Parameters [2].

Parameter	Nominal Value
E-mode Threshold Voltage	+ 0.150 V
E Imax (Vds=1.5V, Vgs≈0.7V)	90 uA/um
E Cutoff Frequency (Ft @ Imax/2)	18 GHz
E Transconductance	225 mS/mm
D-mode Pinchoff Voltage	- 0.600 V
D Idss (Vds=1.5V)	70 uA/um
D Cutoff Frequency (Ft Gm/C)	20 GHz
D Transconductance	200 mS/mm
G (Power FET) Pinchoff Voltage	- 2.20 V
G Idss (Vds=3.0V)	270 uA/um
G Imax (Vds=3.0V, Vgs≈0.7V)	400 uA/um
G Breakdown (Avalanche, Vd=6V)	> 15V
Thin Film Resistor Sheet Resistance	50 Ω/square
G- Implant Resistor Sheet Resistance	600 Ω/square
N+ Implant Resistor Sheet Resistance	100 Ω/square
MIM Capacitance per Area	1.2 fF/um^2
Metal 1 Sheet Resistance	13 mΩ/square
Metal 2 Sheet Resistance	6 mΩ/square

2.2 Components

There are eight supported circuit elements in this process. Supported elements are those that TriQuint has characterized and measure routinely in the PCM. Using structures that are not supported or violating layout rules may have adverse effects on circuit functionality and yield. Supported devices of various sizes are in the device library described in section 6 of [2].

2.2.1 Transistors

E-Mode MESFETs with gate lengths of 0.6 to 20 μm and widths from 2 μm to 250 μm are supported. Single gate and dual gate configurations are supported. These FETs have highly doped channels for high performance small signal and low noise operations. EFETs also feature very low knee voltage and high transconductance for low power supply voltages down to 1V.

D-Mode MESFETs with gate lengths of 0.6 to 20 μm and widths from 2 μm and 250 μm are also supported. Single and dual gate configurations are supported. These FETs are typically used for current sources, switches, small signal and medium power RF operations.

A second depletion MESFET called GFETs with gate lengths of 0.6 to 20 μm and widths from 2 μm and 250 μm are supported. Single and dual gate configurations are supported. These FETs have very high current and power capability. Typically, these FETs are used in very large multi-finger arrangements to make highly efficient power transistors. They are also useful for current sources and high power switches [2].

2.2.2 Diodes

N+ Overlap diodes with anodes (using Schottky gate metal) of 3.0 μm long by widths of 2.0 μm to 250 μm are supported. These are commonly used for level shifting [2].

2.2.3 Resistors

Three resistor types are supported. N+ Implanted Resistors and G- Implanted Resistors have a nominal sheet resistances of 120 Ω/square and 600 Ω/square respectively. They can be sized with widths from 2.0 μm to 250 μm and length from 2.0 μm to any length. These resistors can be used for non-precision uses. Precision NiCr resistors have a nominal sheet resistance of 50 Ω/square and have the same lengths and widths as Implant Resistors. TriQuint's NiCr has a virtually zero temperature coefficient (Tc), so it is ideal for precision applications. Long meander structures can be used to achieve higher resistance values. It is strongly recommended to use NiCr resistors wherever possible since implanted resistors are highly sensitive to backgating and width scaling effects [2].

2.2.4 MIM Capacitors

MIM capacitors are formed with a thin silicon nitride layer between Metal0 and MIM metal layers. MIM capacitors may be sized from a minimum of 5.0 μm per side to any area [2].

2.2.5 Inductors and other Metal Structures

Spiral Inductors and other metal structures, such as strip lines, are easily formed using three metal layers singly or in combination. The metal layers are encased in an Interlayer dielectric constant of about 3.0 [2].

2.3 Interconnects

There are three interconnect layers. The first layer, called Metal0, is formed by evaporation and lift-off of a sputtered 0.5μm layer of Au. This layer is used for local interconnect to contact Ohmic, Gate, and NiCr. It forms the bottom layer of the MIM structure and acts as the contact layer for Metal1.

The other two global layers are made of plated Au for low series resistance interconnect and high Q passives. Metal1 is nominally 2 μm thick and Metal2 layer is nominally 4 μm thick. A thick interlayer dielectric (dielectric constant $\cong$ 3) is between layers of Metal0 and Metal1 and between layers of Metal1 and Metal2. Vias are etched in the dielectric to provide contact between the metal layers above and below the Vias.

For process reasons, vias between Metal0 and Metal1 (the Via1 layer) are restricted in width to preserve planarization during processing. Bond pads use a slotted Via1 structure to accomplish this planarization requirement. Vias between layers may be stacked without special allowances provided that wiring pitch and Via inclusion rules are followed [2].

2.4 Substrate Vias

Substrate Vias are also available on TQTRx.

2.5 Chip and Reticle Dimensions

2.5.1 Chip Thickness

Wafers are initially 25 +/-1 mils thick. Wafers can be thinned to a variety of nominal thicknesses after front end processing. Thicknesses of 20, 10, or 7 mils are standard. If the Die is going into a commercial plastic package then it must be 7 mils [2].

2.5.2 Chip Size and Orientation

The chip size is defined as dicing street center to dicing street center and should be rounded up to the nearest 1 mil (1 mil = 25.4μm). Gates may be oriented <u>either</u> parallel or perpendicular to the flat. The maximum recommended aspect ratio is 2:1 [2].

2.5.3 Reticle Size

The reticle size is defined as the dicing street center to dicing street center and should be rounded up to the nearest 1 mil. The size of the reticle field is a multiple of the chip size. The maximum field size is 580 mils X 580 mils [2].

2.6 Process Control Monitor Size and Placement

PCMs are available in three configurations: Horizontal, Double and Rectangular. When Substrate Vias are used, a Substrate Via Module must be added. There is one PCM per reticle field. The following area needs to be allowed in each reticle field for one of the following PCM sizes:

Table 2.2 [2].

PCM Configuration	PCM + Street Size (mils)
Horizontal	575 X 21
Double	291 X 34
Rectangular	135 X 61
Substrate Via Module	66 X 39 (Additional)

These sizes are correct at the time of printing. These sizes are subject to change [2].

3. MMIC Distributed Amplifier Design Methodology

The principle of distributed (or travelling-wave) amplification using discrete transistors is a technique whereby the gain-bandwidth product of an amplifier can be increased. In this approach, the input and output capacitances of the transistors are combined with lumped inductors to form artificial transmission lines. These lines are coupled by the transconductances of the devices. The amplifier can be designed to give a flat, low-pass response up to very high frequencies. The method used here follows [4] very closely.

Distributed GaAs FET (or MESFET in MMIC area) amplifier designs can provide very flat gain from dc to over 20 GHz – a very significant result. Other technologies above 20 GHz are also available like pHEMT or InP. Distributed amplifiers have been constructed with GaAs MMIC fabrication techniques, because the small size of the MMIC lends itself to the construction of a distributed amplifier. However, a successful distributed amplifier also may be constructed using hybrid techniques. The reason that distributed amplifiers offer outstanding gain-bandwidth products is that they are not matched amplifiers and therefore not subject to Fano's bandwidth limit. A distributed amplifier works on a different concept, called travelling wave amplification.

The topology of the distributed amplifier is particularly suited to Monolithic Microwave Integrated Circuits because its passive circuit predominantly consists of inductors which can be realized in the form of short lengths of microstrip lines. The design of the distributed amplifier involves a careful choice of the variables, such as the device, the number of devices, and the impedances and the cutoff frequencies of the lines, to obtain the desired frequency response. Broadband MMIC distributed amplifiers using GaAs MESFET have been used since 1980's. A schematic representation of the FET distributed amplifiers is shown in Fig.3.1 [4].

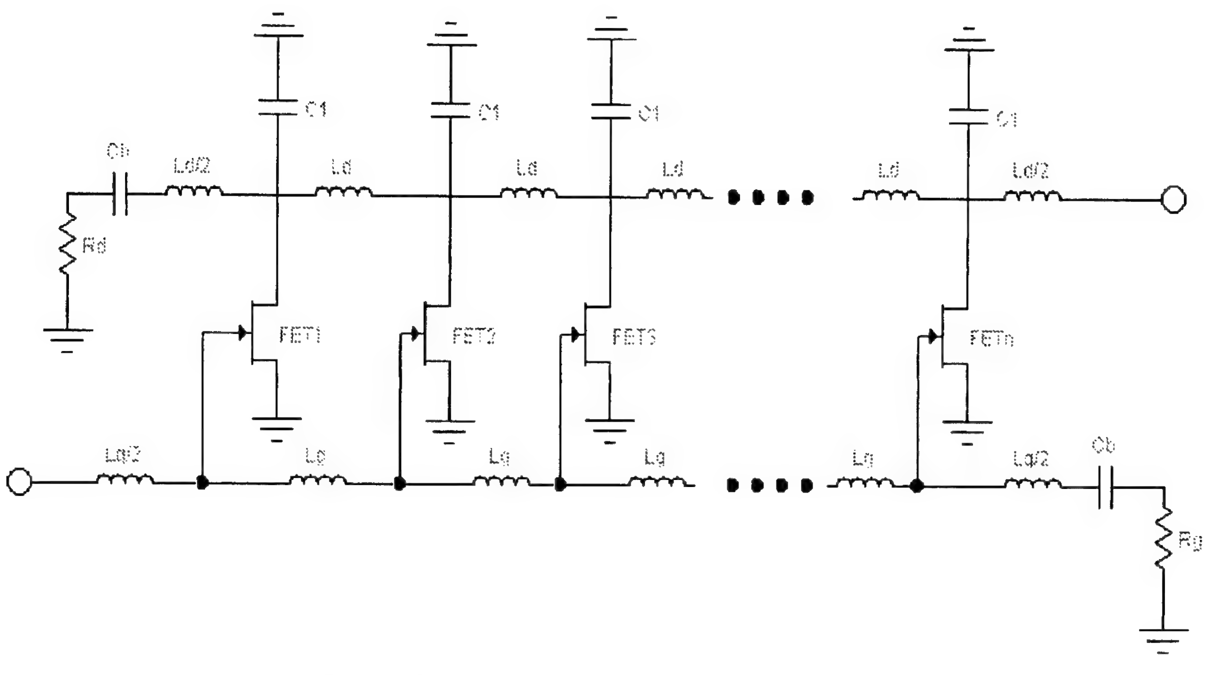


Fig.3.1. Schematic of FET distributed amplifier.

The gate and drain impedances of the FET's are absorbed into the lossy artificial transmission lines formed by using lumped inductors as shown. The resultant transmission lines are referred to as the gate and drain lines. The lines are coupled by the transconductances of the FETs [4].

An RF signal applied at the input end of the gate line travels down the line to the terminated end, where it is absorbed. As the signal travels down the gate line, each transistor is excited by the travelling voltage wave and transfers the signal to the drain line through its transconductance. If the phase velocities of the travelling waves on the gate and drain lines are identical, then the signals on the drain line add in the forward direction as they arrive at the output. The waves travelling in the reverse direction are not in phase, and an uncancelled signal is absorbed by the drain line termination [4].

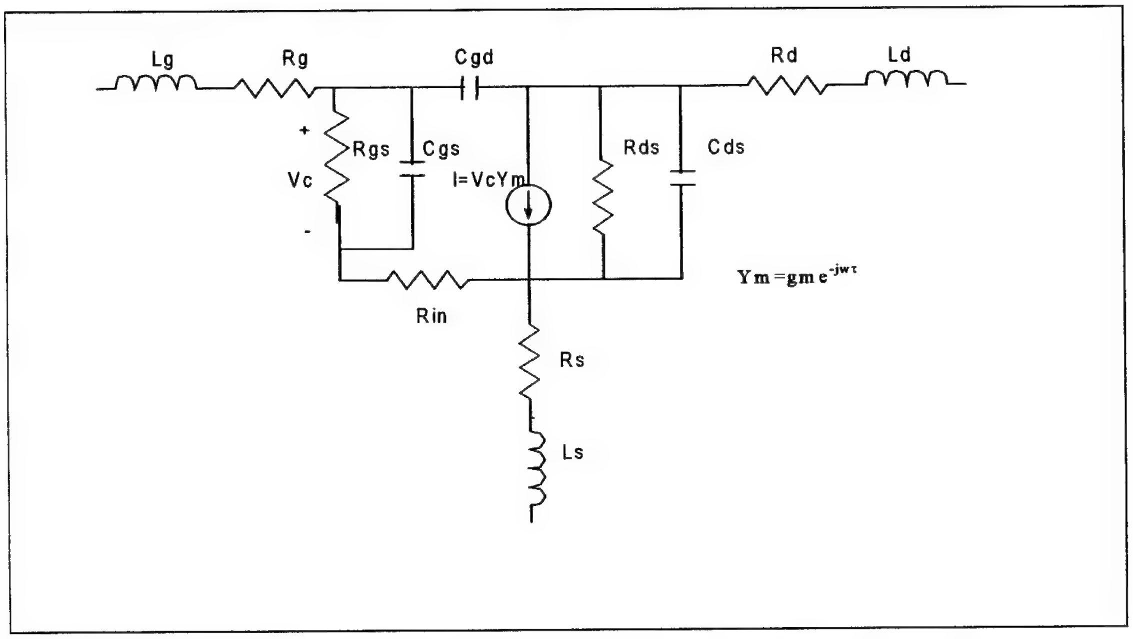


Fig.3.2. Equivalent circuit of a MESFET [2].

An equivalent circuit of a MESFET is shown in Fig.3.2. R_{in} is the effective input resistance between the gate and source terminals and C_{gs} is the gate-to-channel capacitance. R_{ds} and C_{ds} are the drain-to-source resistance and capacitance respectively. C_{gd} is the drain-to-gate capacitance and g_m the transconductance [1]. Here, we will start to elaborate the design equations for designing distributed amplifiers [4].

The shunt capacitances of the gate line are provided by the gate capacitance, C_{GS}, of the FETs. The shunt capacitors of the drain line are provided by the FET's drain capacitance, C_{DS}, plus the supplemental value, C_1.

The characteristic impedance of the gate line and the drain line are equal and given by

$$Z_0 = \sqrt{L_G / C_{GS}} = \sqrt{L_D / (C_{DS} + C_1)} \qquad (3.1)$$

where L_G and L_D are the inductances per section of the gate line and the drain line, respectively. To produce useful gain, it is very important that the drain currents generated by each individual FET add in-phase as the signal moves along the drain line toward the amplifier's output. Therefore the phase shift between FETs along the drain line must be exactly the same as the phase shift between FETs along the gate line [4]. The phase shift of a single section of artificial transmission line is given by

$$\Phi_G = 2\pi f \tau_G \tag{3.2}$$

in the gate line, and by

$$\Phi_D = 2\pi f \tau_D \tag{3.3}$$

in the drain line where τ_G is the gate line time delay per section, and τ_D is the drain line time delay per section. The condition that will ensure constructive interference of all FET currents, at all frequencies, is

$$\Phi_G = \Phi_D \tag{3.4}$$

or

$$\tau_G = \tau_D \tag{3.5}$$

Now, for any artificial transmission line, the delay per section is equal to $\sqrt{LC}$; therefore equation (3.5) can be written as

$$\sqrt{L_G C_{GS}} = \sqrt{L_D (C_{DS} + C_1)} \tag{3.6}$$

For a realistic amplifier, the impedance conditions, (3.1), and the phase conditions (3.6) must be satisfied simultaneously. Simultaneous solution is given by

$$L_G = L_D = L \tag{3.7}$$

and

$$C_{GS} = C_{DS} + C_1 \tag{3.8}$$

or

$$C_1 = C_{GS} - C_{DS} \tag{3.9}$$

A third condition that must be imposed if the amplifier is to work in a 50 Ω system is

$$Z_0 = 50 = \sqrt{L/C_{GS}} \tag{3.10}$$

or

$$L = 50^2 C_{GS} \tag{3.11}$$

We are in a position to calculate the gain and bandwidth of a distributed amplifier. Ideally the bandwidth will be from dc to a frequency a little less than the cut-off frequency, f_c, of the gate and drain artificial lines. The cut-off frequency is given by

$$f_c = \frac{1}{\pi Z_0 C_{GS}} \tag{3.12}$$

The current out of each FET adds in-phase along the drain line and is divided equally between the output and the termination. The gain of a distributed amplifier (neglecting loss) is very simply one half of the output current per FET times the number of FETs, times the load resistance, divided by the input voltage:

$$G = V_{out}/V_{in} = \frac{ng_m V_i Z_0 / 2}{V_i} \tag{3.13}$$

or, in decibels

$$G = 20\log_{10}\left(\frac{ng_m Z_0}{2}\right) \tag{3.14}$$

where

n is the number of sections (same as the number of FETs)

Z_0 is the input and output characteristic impedance (50 Ω)

g_m is the transconductance of an individual FET

Equations (3.12) and (3.14) are elegantly simple expressions for the cut-off frequency and the gain of a distributed amplifier. Multiplying these expressions together yields the *maximum gain-bandwidth* (GBW) product of a distributed amplifier:

$$GBW = \left(\frac{1}{\pi Z_0 C_{GS}}\right)\left(\frac{nZ_0 g_m}{2}\right) = \frac{ng_m}{2\pi C_{GS}} \tag{3.15}$$

The cut-off frequency of the individual FETs, f_t , is

$$f_t = \frac{g_m}{2\pi C_{GS}} \tag{3.16}$$

Therefore,

$$GBW = nf_t \tag{3.17}$$

This important result simply states that the maximum gain-bandwidth product of an ideal distributed amplifier is equal to the cut-off frequency of its individual FETs times the number of FETs [4].

Gain and bandwidth may be traded off by changing the gate width, w, of the individual FETs. For a high cut-off frequency, use a small C_{GS}, corresponding to a small gate width. Under these conditions, n must be raised to build up the gain, because g_m will be low in the individual FETs. This is why high-frequency distributed amplifiers use many small FETs. The value n=6 or 7 is not unheard of in amplifiers operating to 20 GHz or more. At low frequencies only a few FETs are used, but they are proportionally larger [4].

To predict the correct size FET for an arbitrary cut-off frequency and gain, the concept of FET equivalent circuit parameters normalized to width is introduced [4]. Define these parameters as follows:

$$g_m = wg_m' \tag{3.18}$$

$$C_{GS} = wC'_{GS} \tag{3.19}$$

$$I_{DSS} = wI'_{DSS} \tag{3.20}$$

where

g'_m is the FET's transconductance per unit width

C'_{GS} is the FET's gate capacitance per unit width

I'_{DSS} is the FET's saturated drain current per unit width

Using the definitions in (3.18), (3.19), (3.20) and (3.12), we can rewrite the amplifier's cut-off frequency as

$$f_c = \frac{1}{\pi Z_0 wC'_{GS}} \tag{3.21}$$

or

$$w = \frac{1}{\pi Z_0 C'_{GS} f_c} \tag{3.22}$$

The individual cut-off frequency is independent of FET width because

$$f_t = \frac{g_m}{2\pi C_{GS}} = \frac{wg'_m}{2\pi wC'_{GS}} = \frac{g'_m}{2\pi C'_{GS}} \tag{3.23}$$

Therefore, the maximum gain-bandwidth product always is given by (3.17) as

$$GBW = Gf_c = nf_t \tag{3.24}$$

which is independent of gate width. In terms of decibels, gain is equal to

$$G = 20\log_{10}\left(\frac{nf_t}{f_c}\right) \tag{3.25}$$

The value of the inductance in the artificial distributed lines may also be calculated from the cut-off frequency by going back to the expression for impedance (3.1):

$$Z_0 = \sqrt{L/C_{GS}} = \sqrt{L/wC'_{GS}}$$

or

$$Z_0^2 = \frac{L}{wC'_{GS}} \tag{3.26}$$

Combining with (3.22)

$$Z_0^2 = \frac{\pi Z_0 C'_{GS} f_c L}{C'_{GS}} \qquad (3.27)$$

After canceling Z_0 and C_{GS}' , and solving for L, we have

$$L = \frac{Z_0}{\pi f_c} \qquad (3.28)$$

The dc current of any distributed amplifier may be calculated in a similar way. Assume that the FET's are to be biased at

$$I_{DS} = \frac{I_{DSS}}{2} \qquad (3.29)$$

The drain-to-source current of an individual FET is

$$I_{DS} = \frac{wI'_{DSS}}{2} \qquad (3.30)$$

For the entire amplifier,

$$I_{dc} = nI_{DS} = \frac{nwI'_{DSS}}{2} \qquad (3.31)$$

From (3.17),

$$Gf_c = nf_t \qquad (3.32)$$

or

$$n = \frac{Gf}{f_t} \qquad (3.33)$$

Combining with (3.22) and (3.23) yields

$$I_{dc} = G\frac{I'_{DSS}}{Z_o g'_m} \qquad (3.34)$$

Following the above steps, one can start to design a distributed amplifier with the design manual of the foundry that will be used in hand. The first movement should be the choice of proper biasing condition (which defines a set of model parameters for the Field Effect Transistors). The set of model parameters given for a certain bias condition are inserted into the linear FET model of a powerful simulator such as HP EEsof Series IV RF simulator with proper choice of drain and gate line inductances and other lumped element components. If the optimization does not give results close to the design specifications different set of model parameters for the transistor models and lumped components should be tried. This iterative procedure is continued until no further parameter is left to choose to improve the performance of the circuit. A mature understanding of above concepts should be obtained to handle the design comfortably.

4. Design Steps and Results

4.1 (3.1-3.5 GHz) Distributed MMIC Amplifier Design Details

Steps included in a general MMIC design is as shown in Fig.4.1. More detailed design guidelines are given in [3]. Our design uses the algorithmic design approach summarized in Fig.4.1.

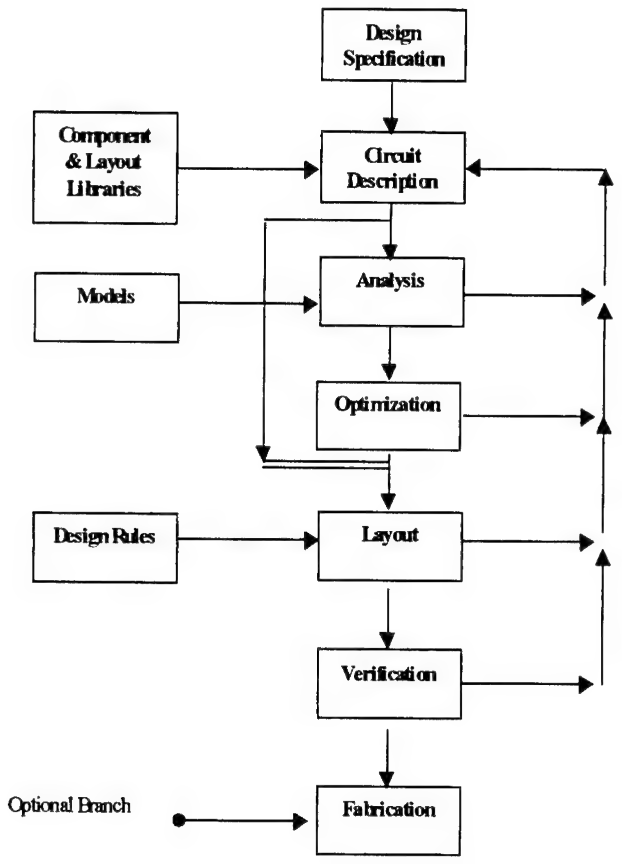


Fig.4.1. MMIC Design Process.

4.2 Design Specifications

Frequency Band	: 3.1 GHz-3.5 GHz	Noise Figure	: < 6 dB
Small Signal Gain	: 12 dB	Stability	: Unconditionally stable
Input Reflection Coefficient	: < -10 dB	Technology	: GaAs MMIC
Output Reflection Coefficient	: < -10 dB	Gain Flatness	: ± 0.5 dB
		Amplification Method	: Distributed Amplification

Of the above specifications, the gain flatness is the most important one. The reason for choosing the distributed amplification method from among some other methods is that one can design a high-bandwidth distributed amplifier with acceptable gain variation in the band of interest and choose a portion of the band where gain variation is small. Noise figure is the second most important requirement to be realized since the output of the system will feed an optical system. Low input and output reflections guarantee that the power loss due to mismatches is minimized. In addition, the device should be highly robust in terms of stability.

4.3 Circuit Description

The circuit schematic generated in HP EEsof Series IV is as in Fig.4.2.

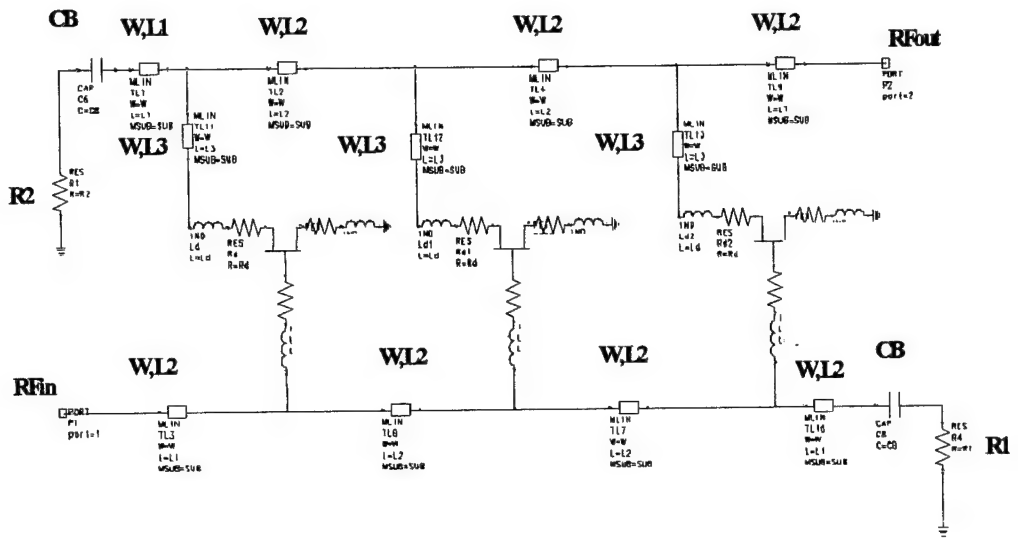


Fig.4.2. Circuit Diagram of the Three-Stage Distributed Amplifier Using Transmission Lines.

Labels W,Li (i=1,2,3) over each transmission line shows the width and the length of the transmission lines, respectively. Transmision lines are microstrip and substrate model is given as follows. Capacitors labeled as CB are drain and gate line MIM (metal-insulator-metal) blocking capacitors. Their value is chosen to be high such that they will be open at DC and almost short at the frequency band of interest. Resistors R1 and R2 are NiCr resistances with 50 Ω values. Each dotted rectangle around the transistors includes the complete linear FET model given in the design manual of the foundry. Table 4.1 summarizes all the component values and physical dimensions of the transmission lines used in the circuit.

Table 4.1. Circuit Component Parameters for the Distributed Amplifier.

Parameter Name	Dimension
W (transmission line widths)	12 μm
L1 (transmission line type #1 length)	1354 μm
L2 (transmission line type #2 length)	937 μm
L3 (transmission line type #3 length)	555 μm
CB (MIM blocking capacitance)	10 pF
R1 (NiCr termination resistance)	50 Ω
R2 (NiCr termination resistance)	50 Ω

Transmission line substrate media parameters are shown in Table 4.2.

Table 4.2 Substrate media parameters.

Parameter	Name	Value
ε_r	Dielectric constant	12.9
H	Substrate thickness	101.6 μm
T	Metal thickness	2μm

The lumped elements shown at the terminals of each FET are included to complete the model of the simulator to the model supplied by the foundry. Transistor model parameters together with the lumped elements are shown in Table 4.3.

Table 4.3. Transistor model parameters .

Parameter Name	Value
V_{ds}	3.0 V
V_{gs}	0.6 V
I_{ds}	20.1 Ma
C_{gs}	0.7455 pF
C_{ds}	0.0778 pF
C_{gd}	0.0382 pF
C_{dc}	0 pF
R_{gs}	408.16 Ω
G_{gs}	$1/R_{gs}$
R_{ds}	338.10 Ω
g_m	0.082360 S
tau	3.815 Ps
R_{in}	0.9887 Ω
R_s	1.096 Ω
R_d	0.1354 Ω
R_g	3.6 Ω
L_s	0 nH
L_d	0.0127 nH
L_g	0.0104 nH

4.4 Analysis and Optimization

With the theoretical knowledge discussed in section 3 of this report, design parameters are started with some rough parameters for the transistors and other circuit components. Linear FET model (see Fig.3.2) parameters is the basic requirement to start the design. In MIC (Microwave integrated circuits), the designer does not have much freedom of using lumped components while a MMIC designer can go either with lumped or distributed elements. Since distributed element approach gives smoother gain and reflection responses and lumped elements are only good in a certain range of frequencies, distributed element design approach is adopted in this design. The term distributed should not be confused with the distributed amplification where the latter is just an amplification method. Transmission lines in MMIC circuits are just pieces of metal stripes with a certain length and width on top of a dielectric substrate (see Fig.4.2). The thickness and dielectric properties of the substrate depend on what foundry one works with and what process of that foundry is used. Therefore, working with different foundries one gets different circuits functioning in the same manner.

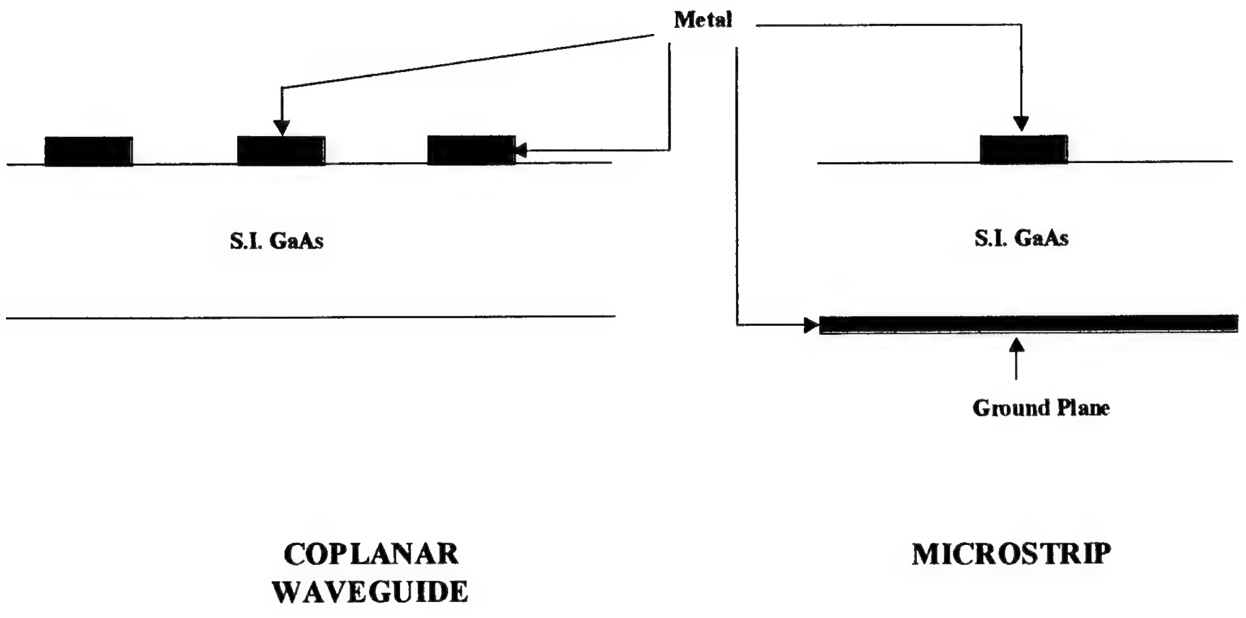


Fig.4.2. Microstrip Transmission Lines in a MMIC Device [4].

The impedance of the line is determined by the ratio of the conductor width to the substrate thickness. An impedance of 50 Ω is generally preferred for input and output transmission lines. The transmission line must be sufficiently thick to carry the required current without significant ohmic losses (at least two skin depths thick). If the lines are to carry DC current, the line widths should be sufficient to support the current density.

The first step in the design was to estimate the lengths of transmission lines used in the distributed amplifiers using the theoretical approach described in Section 3 of this report. In the design manual supplied by the TriQuint semiconductor, there are some premeasured linear FET models tabulated for different bias conditions. The simulation program (HP EEsof Series IV) contains a linear FET model that is very close to the exact linear FET models supplied by TriQuint. The only difference is the contact resistances and inductances. These additional model parameters are included in the design by adding lumped inductors and resistors in the terminals of the FETs as having the same values given in the model of TriQuint.

After the primary simulation results were obtained in the band of interest, the optimization tool of the simulator is used to get the required response. The optimization criteria are given in Table 4.4.

Table 4.4. Optimization criteria

Optimization Variable	Optimization Goal	Optimization Weight
NF (dB)	< 3 (dB)	10
S21(dB)	= 12 (dB)	80
S11(dB)	< -10 (dB)	10
S21(dB)	< -10 (dB)	10

The optimizable variables in the design are transistors parameters (restricted to the values given by the foundry), the width and the length of the transmission lines. The width of the transmission lines is set to a minimum such that there will be no problem in carrying the DC current . Trying different operation modes and different bias conditions does not affect the performance very much so it is not also practical to optimize the model parameters of the transistors. The only optimizable parameter is , therefore, the length of the transmission lines. The lengths of the transmission lines cannot be lower than 7 µm under the given bias conditions.

There are also some other constraints, the most important of which is the length of the transmission lines in between the drains and gates of the transistors. In the optimization, it is considered that the transistor layout that will replace the model of HP EEsof Series IV is a 6 finger FET layout (each finger has 50 µm width) with a rectangular shape. To be able to interconnect the drain and gate points of the transistors, the lengths of the transmission lines should not be less than a minimum which depends on the transistor size. This brings a lower limit to the transmission line lengths that are the only optimizable variables in the design. As can be seen from the above discussion, a MMIC designer has to deal with many parameters overlapping with other phases of the design.

4.5 Simulation Results

During the design, different sets of transistor model data from the design manual are tried to get as much close as possible to the desired response. If the bias conditions under which the model parameters are obtained are satisfied in the band of interest, circuit must function in well agreement with the design specifications. After several runs of the optimization tool, responses of the design are obtained as in the following figures.

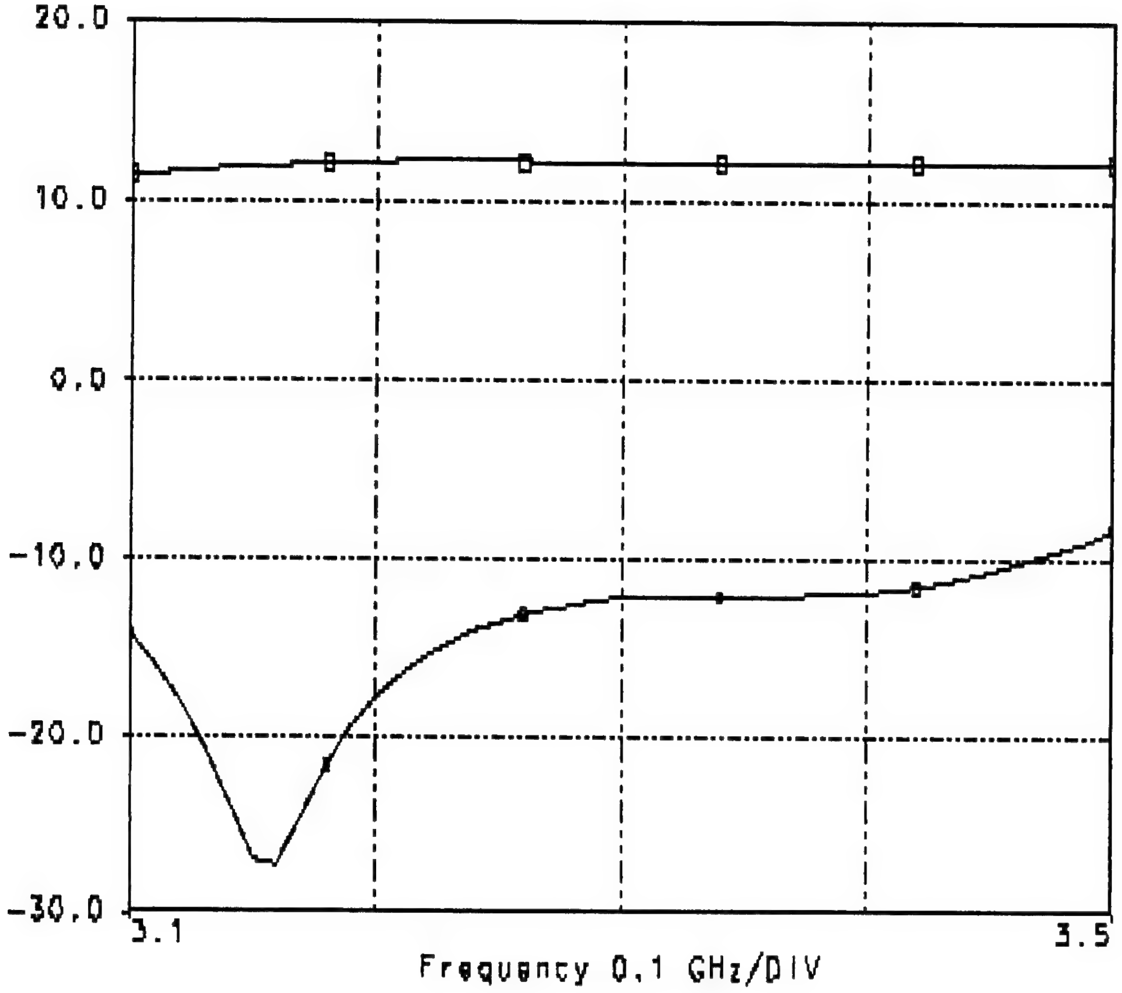


Fig.4.3. Small signal gain and input reflection coefficient in dB.

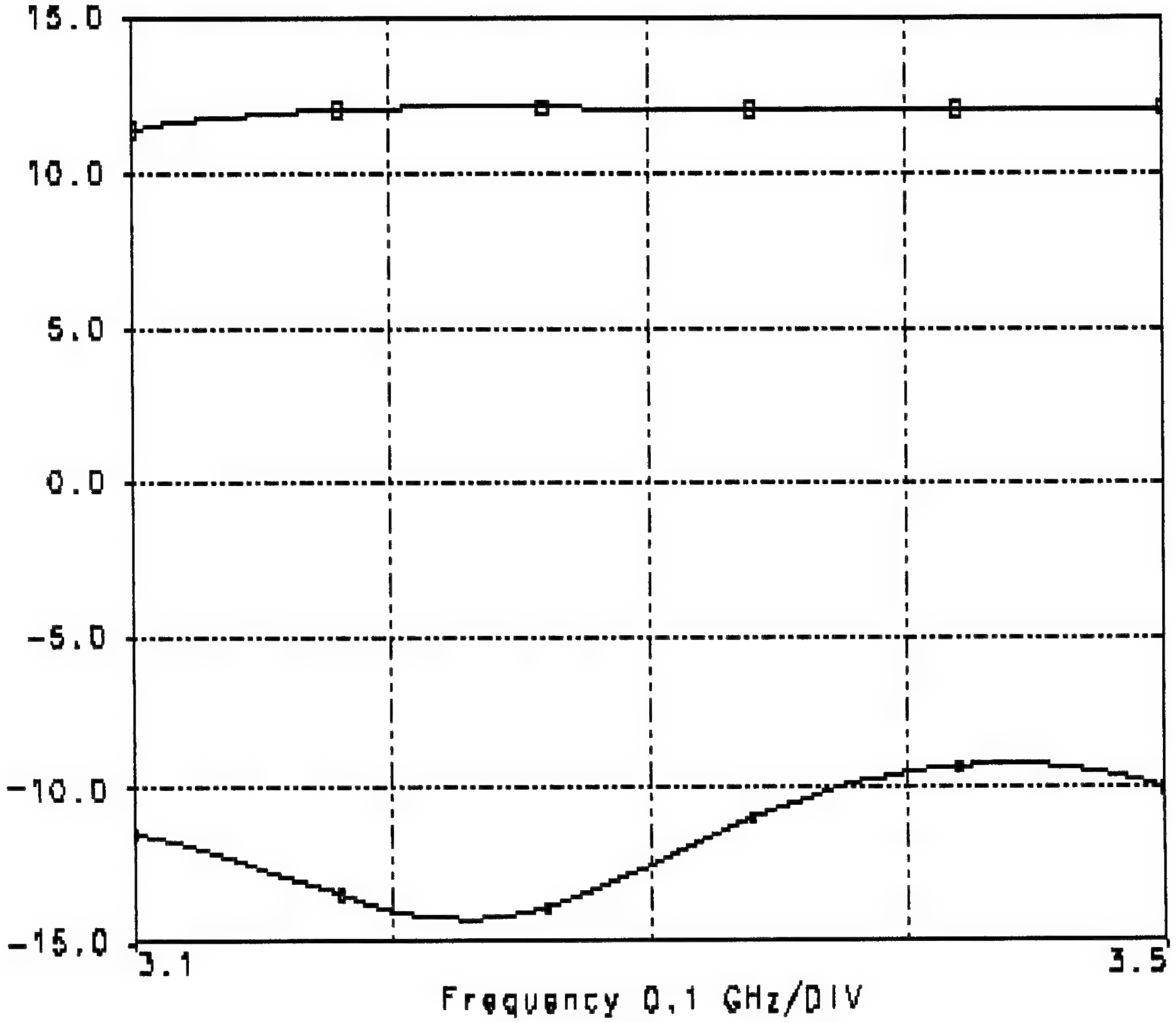


Fig.4.4. Small signal gain and output reflection coefficient in dB.

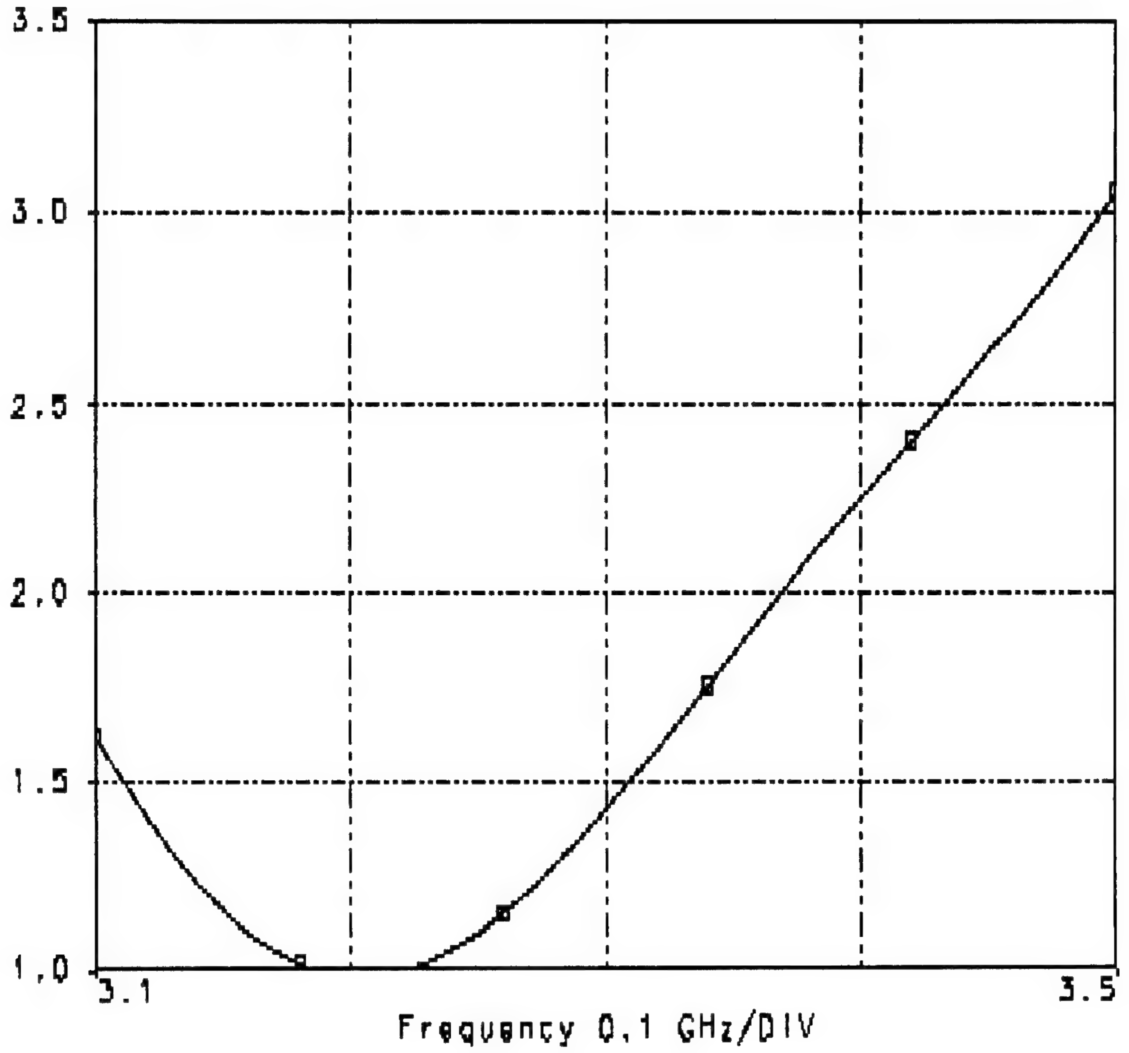


Fig.4.5. Noise Figure in dB.

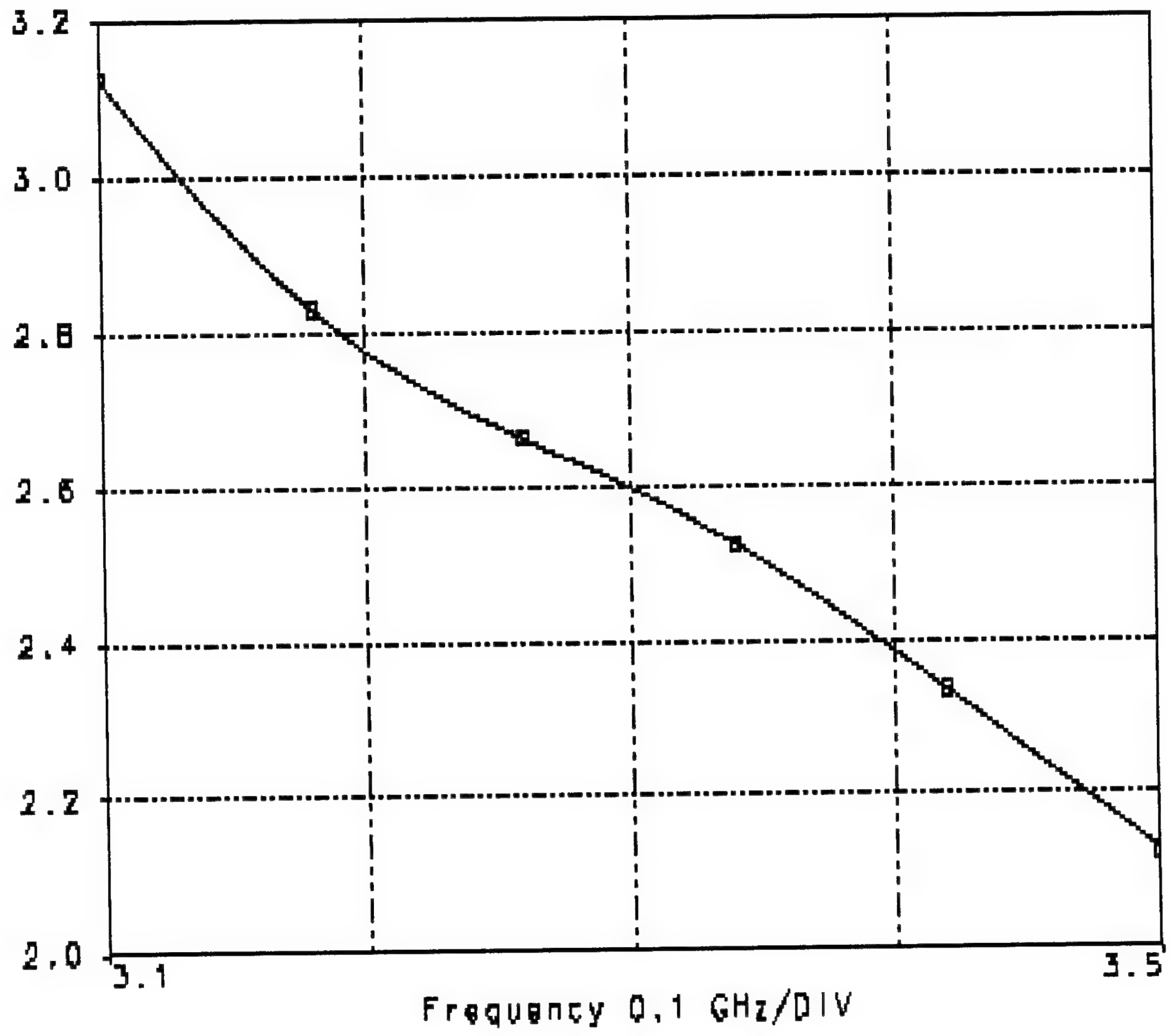


Fig.4.6. Stability Constant K.

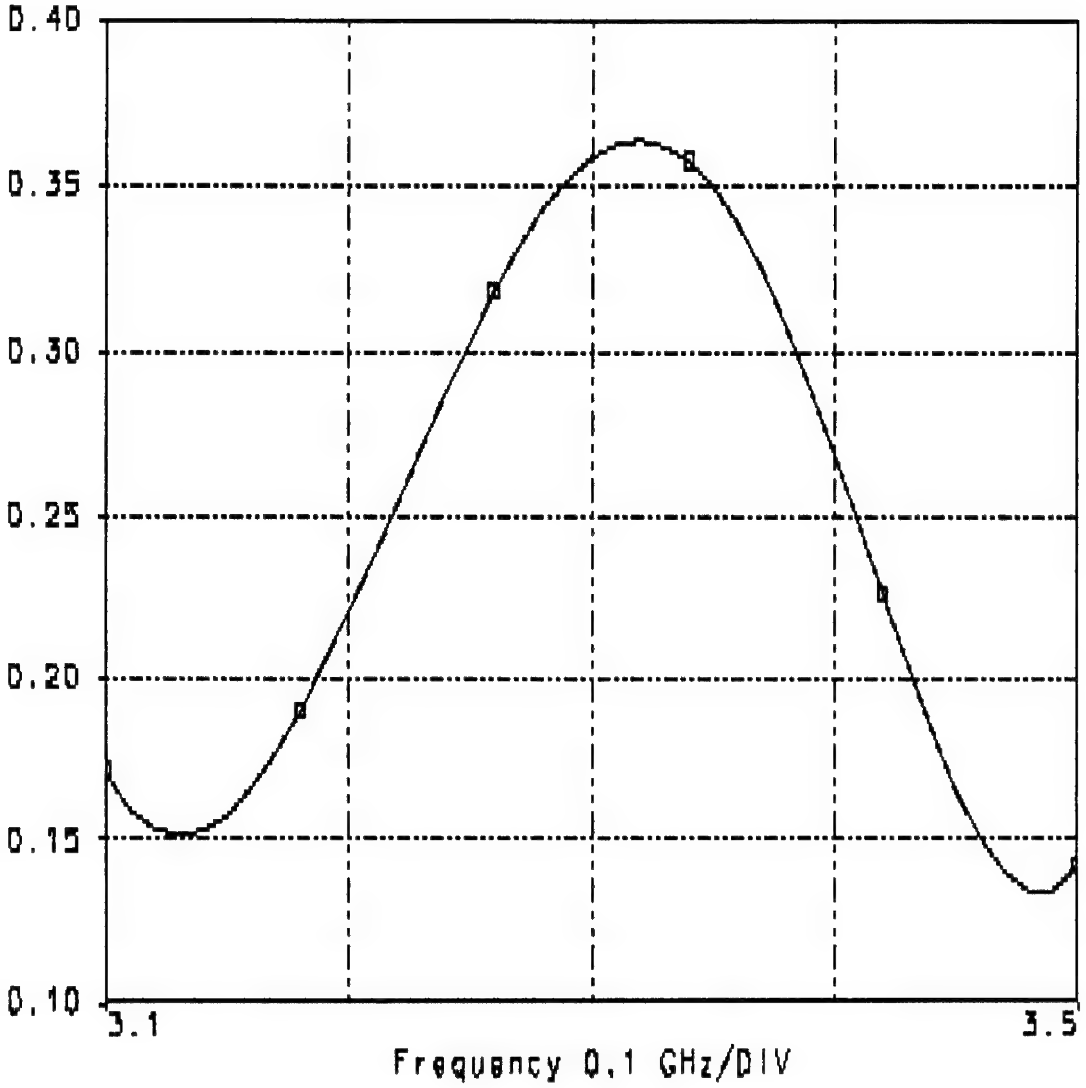


Fig.4.7. Noise Resistance in Ω.

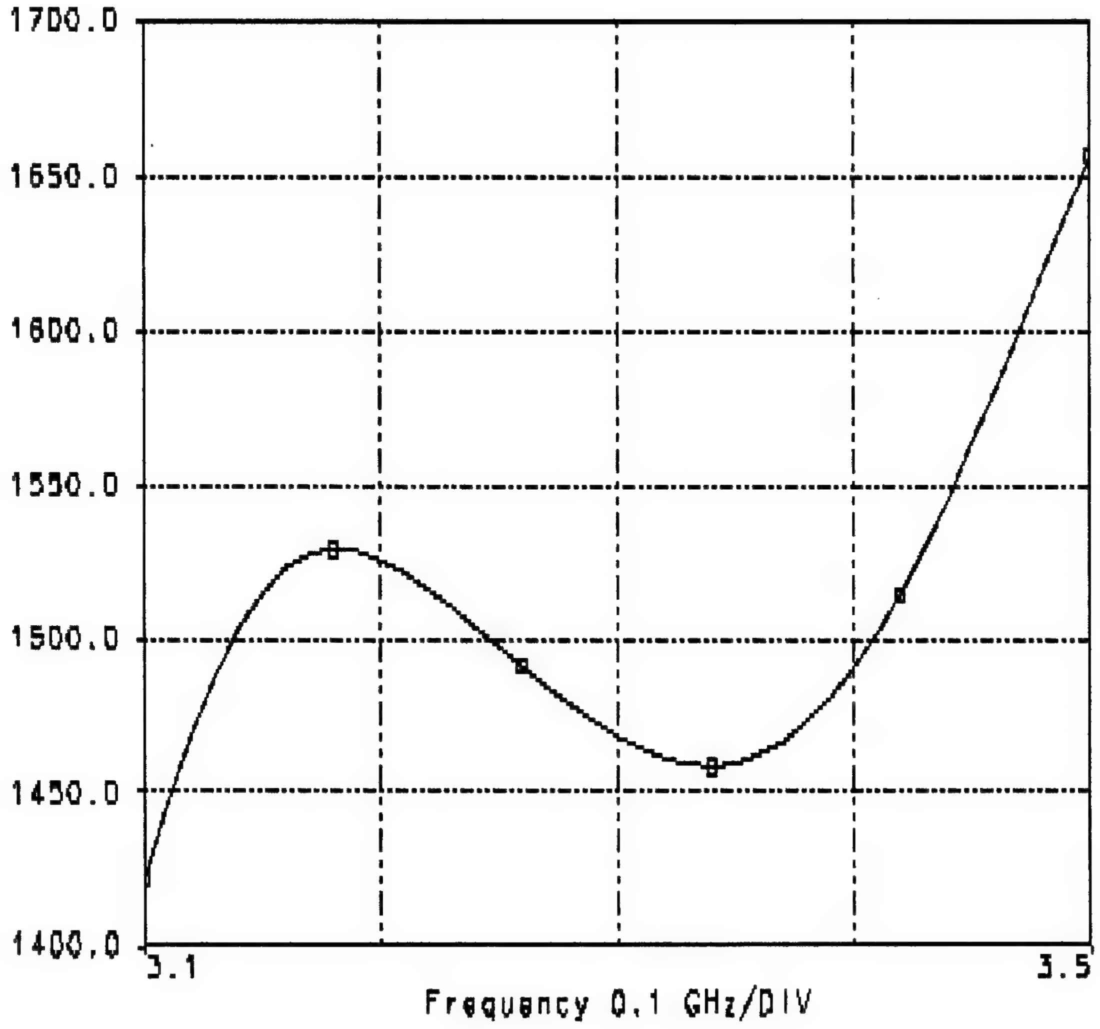


Fig.4.8. Group Delay in ps.

As it can be seen from the figures, the responses are in very good agreement with the design specifications. The next phase is to generate the layout of this device on a GaAs substrate using a software called ICED. Before going into the layout phase the bias circuitry will be discussed shortly.

4.6 Bias Circuitry

As seen in Figure 3.1, both the gate line and the drain line circuits are DC isolated from their terminations by a blocking capacitor, C_B. A good way to bias a distributed amplifier is to use two DC supplies (i.e., a positive supply for the drain circuit and another bias for the gate circuit). A technique must be developed for applying these voltages to their respective lines without disturbing the RF performance. A very straightforward technique for doing this is to provide off-chip inductive bias chokes connected directly to the amplifier's input and the output terminals, as shown in Fig. 4.9.

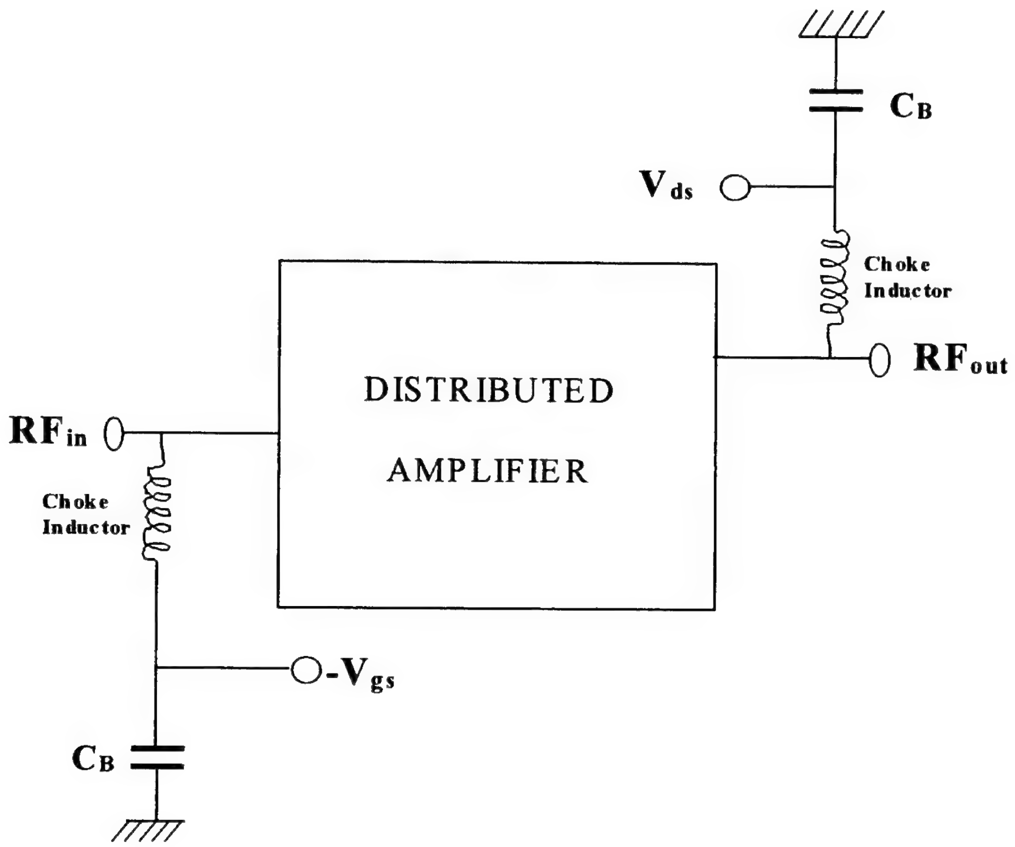


Fig.4.9. Simple off-chip bias network for the MMIC distributed amplifier [4].

An alternative on-chip bias scheme would involve placing both a gate bias choke and a drain bias choke on the chip. These chokes would be spiral inductors. The reactance of such devices must be at least 100 Ω at 3.3 GHz. Their inductance is calculated as

$$L_C = \frac{100}{2\pi(3.1.10^9)} = 5.134\,nH$$

To design a 5.13 nH spiral inductor that is resonance free over the entire 3.1 to 3.5 GHz band is not very difficult. Since this project is a prototype for the real device, the off-chip-biasing scheme, which is the easier one, is preferred. With this bias scheme, it is possible to play with the bias components in case of unsatisfactory responses. For the real design, an on-chip bias scheme should be preferred since no bulky bias circuitry is acceptable in the application area of this device.

4.7 Layout

In this phase of the project, the layout editor called ICED (IC Editors, Inc.) is utilized to generate the GDSII stream file to be sent to the foundry to be manufactured. The TQTRx device library is available in a GDSII STREAM format file. Some subcells previously generated by the engineers of the foundry are supplied with the design manual of the foundry process. In the layout, following subcells are used.

xHSABFET	**: FET, Multi-finger FET, Width=6 X 50 μm, RF Probe Pads**
C25X25	**: 25 μm x 25μm effective area, C ≅ 0.75 Pf**
RN10	**: NiCr resistor, L=10 μm, W=10 μm**

The elements in the device library of the foundry are used as templates. The size of the elements is stretched according to the rules given in the design manual. Violating these rules can have very adverse effects on circuit functionality and yield. TriQuint provides a design rule check (DRC) rule set for IC Editors "The DRC". This low cost software for PC is used in conjunction with IC Editors ICED-32^{TM} Layout Editor.

xHSABFET subcell is used as it is while the other two subcells are modified to get the values in the circuit file. Final layout of the circuit is shown in Fig. 4.10. The manufactured piece has arrived at the end of the project duration leaving no time for testing. We plan to perform measurements on the piece and compare the real performance with the simulation results.

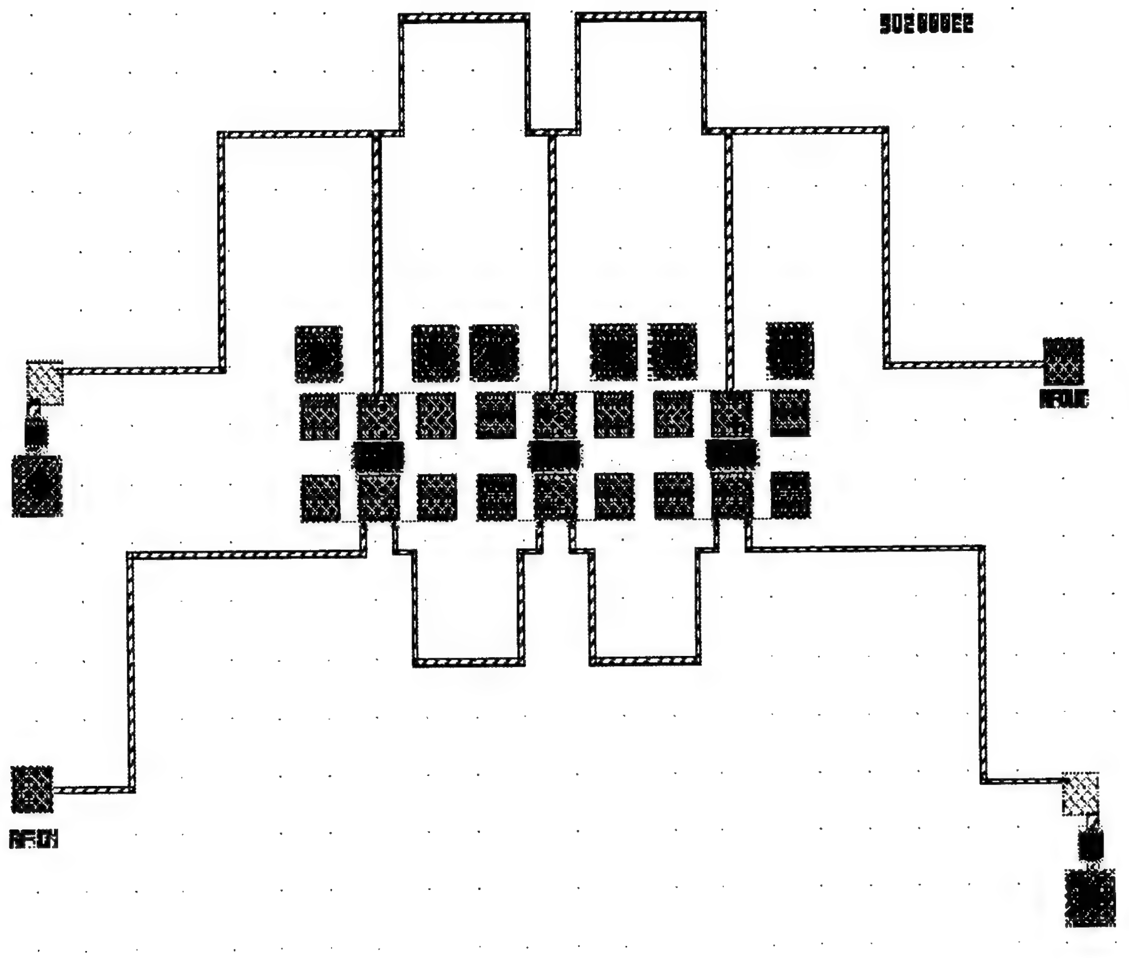


Fig.4.10. Layout of the MMIC Distributed Amplifier.

5. References

[1] James B. Beyer, et.al., " MESFET Distributed Amplifier Design Guidelines" IEEE Trans. Microwave Theory Tech., vol. MTT32, PP. 268-275, March 1984.

[2] GaAs IC Design Manual, Triquint Semiconductor, 1999.

[3] Design Guidelines for Microwave Monolithic Integrated Circuits, Author: G.Gatti (XRM), Reference XRM/017.95/GG, Issue 3 Rev.0, Dated 25 Sept. 1995.

[4] MIC & MMIC Amplifier and Oscillator Circuit Design, Alan Sweet, Artech House, Boston-London,1990.

[5] GaAs IC Design Class Manual, Triquint Semiconductor,1999.

6. Acknowledgements

I wish to thank AFOSR for the SREP and to Dr. Richard Michalak and Mr. Michael Little of Rome Lab for various discussions and inputs. The problem addressed here was an excellent example that was used as a class project in educating my students taking the "CAD for Microwave Circuits" course. Special thanks are to my student Mr. Ertan Zencir who performed most of the simulation work as his class project. I also want to thank the other students in the class for many valuable discussions and inputs. I want to thank HP and Ic Editors for using their software in our class and TriQuint for manufacturing the amplifier.

DETECTION OF ACOUSTIC CORRELATES OF STRESS FROM THE MODULATION CHARACTERISTICS OF SPEECH

Kaliappan Gopalan

Professor

Department of Engineering

Purdue University Calumet

Hammond, IN 46323

Final Report for:

Summer Research Extension Program

Air Force Research Laboratory

Rome Research Site

Sponsored by:

Air Force Office of Scientific Research

Bolling Air Force Base, DC

AFRL, Rome Research Site

and

Purdue University Calumet

January 2000

DETECTION OF ACOUSTIC CORRELATES OF STRESS FROM THE MODULATION CHARACTERISTICS OF SPEECH

Kaliappan Gopalan
Professor
Department of Engineering
Purdue University Calumet
Hammond, IN 46323

Abstract

The effect of workplace stress on certain speech parameters of the AM-FM speech model was studied in this project. Correlation between demodulated amplitude and frequency based parameters and the heart rate of a fighter aircraft flight controller was investigated. It was found that the peak frequencies in the spectrum of the amplitude envelope (AE) in the AM-FM model followed F0, the fundamental frequency of voicing, regardless of the center frequency of analysis. This tracking of F0 gives a qualitative estimate of stress level relative to a neutral state of stress with no a priori knowledge of F0 or formants. Additionally, the mean and median values of F0 estimates from the spectra of the AE were found higher at lower or higher heart rates than neutral rate. Formants estimated from the instantaneous frequency, on the other hand, increased or decreased with heart rates depending on their indices - F3 decreased with heart rate while F4 increased. Similar results were observed from the utterances of an F-16 pilot in an aviation emergency.

DETECTION OF ACOUSTIC CORRELATES OF STRESS FROM THE MODULATION CHARACTERISTICS OF SPEECH

Kaliappan Gopalan

I. Introduction

Analysis of physiological stress of speakers based on their speech utterances is important in many applications. Response by an emergency management team, for example, may be tailored to the needs of the caller based on the level of stress detected from the caller's speech. Detection of physiological stress at a workplace such as an aircraft is important in assessing the ability of the worker and assigning tasks accordingly. Stress detection based on speech enables monitoring of stress nonintrusively and, in general, without the cooperation of the speaker. Other applications include detection of deception in speech for law enforcement use based on emotional state, and implementation of natural-sounding speech synthesizers.

This report presents the results of analyzing certain modulation characteristics as a function of the heart rate of fighter aircraft pilots. Additionally, the variation of modulation features as a pilot undergoes an operational emergency is discussed.

II. Modulation Behavior in Speech

Motivation for the analysis of speech using amplitude and frequency modulations was provided by the observation by Teager and Teager [1, 2] that modulation process dominates the production of speech. Teager and Teager showed that in addition to the frequency modulation causing instantaneous frequency variations, speech resonances also have time-varying amplitudes. Other researchers postulated that human auditory system uses transduction of frequency modulation (FM) to amplitude modulation (AM) using the spectral shapes of auditory filters [3]. Based on the results of these works, the many nonlinear and time-varying phenomena during speech production have been modeled successfully by an AM-FM modulation model representing each of the resonances or formants [4, 5].

In the case of a speaker under stress, increased activation of the sympathetic nervous system or the parasympathetic nervous system is observed to occur when the speaker is angry, fearful, sad, etc [6]. This increased activation leads to change in heart rate and blood pressure, and also to tremor in muscle activity. Consequently, the articulatory and respiratory movements for speech production are affected. The spectral characteristics of the resulting stressed speech are, in general, observed to have increased fundamental frequency F0, increased amplitude, and decreased speech duration. The extent of variation in F0, however, has been shown to depend on the speaker and the type of stress. Other manifestations of stress in speech include variations in the formants and their bandwidths, increase in high frequency energy, and changes in the glottal pulse shape.

In addition, studies have shown that in stressed speech, the fundamental frequency of excitation and formants have higher variations in their instantaneous values than in neutral speech. Therefore, an analysis of the variation of F0 and the formants is expected to bring out the frequency variations due to stress. Maragos et al [7] observed that the time-varying instantaneous frequencies and amplitudes can be extracted from the multicomponent AM-FM models.

The AM-FM model represents speech around each formant F_k as a damped AM-FM signal given by [4, 7]

$$x(t) = \sum_{k=1}^{N} a_k(t)\cos\left(\omega_k t + \Phi_k(t)\right) \qquad (1)$$

where N represents the number of formants in the speech signal x(t). $a_k(t)$ is the time-varying amplitude of the sinusoid at the kth formant frequency ω_k with the total instantaneous frequency of $\omega_i = \omega_k + \frac{d\Phi_k}{dt}$. Both the instantaneous frequency (IF) and the amplitude envelope (AE) around a formant are derived from the Teager energy operator [1, 2]. Because of the unlimited number of combinations of AM and FM that can give rise to the modulated signal at each formant, the modulation model given in (1) must be analyzed separately at each formant.

Representing the discrete-time version of a speech signal band-pass filtered at the center frequency $f_c = \omega_c/2\pi$ by

$$s(n) = a(n)\cos\Phi(n)$$

or

$$s(n) = a(n)\cos(\Omega_c n + \Omega_m \int_0^n q(k)dk), \qquad (2)$$

the instantaneous frequency Ω_i given by

$$\Omega_i(n) = \frac{d\Phi(n)}{dn} = \Omega_c + \Omega_m q(n) \qquad (3)$$

where Ω_m is the maximum frequency deviation from Ω_c.

One of the most commonly used methods for obtaining the IF and AE employs the nonlinear energy tracking operator devised by Teager [1, 2]. Using the algorithm formulated by Kaiser, the Teager energy of a discrete signal s(n) is calculated by the operator $\Psi(.)$ given by [8]

$$\psi[s(n)] = s^2(n) - s(n-1)s(n+1) \tag{4}$$

From the energy operators of a speech signal and its time-shifted versions, the instantaneous frequency Ω_i and the amplitude envelope $|a(n)|$ are calculated using the energy separation algorithm [5, 7] as

$$\Omega_i(n) \approx \arcsin\sqrt{\frac{\psi[s(n+1) - s(n-1)]}{4\psi[s(n)]}} \tag{5}$$

$$|a(n)| \approx \frac{2\psi[s(n)]}{\sqrt{\psi[s(n+1)] - \psi[s(n-1)]}} \tag{6}$$

Preliminary studies of the IF and AE showed that features based on the above AM-FM demodulation model may indicate variations in the emotional stress of the speaker [9]. In particular, the spectrum of AE showed higher peak frequencies at higher levels of stress; additionally, the spectra of both IF and AE, in general, followed the fundamental frequency F0. Since F0 is a significant indicator of stress [10-13], an estimation of F0 from the demodulation features is expected to provide a correlation with stress. Moreover, demodulation features leading to F0 estimation – such as the bandwidth of IF around the center frequency Ω_c, jitter in F0, etc. - may also vary in proportion to stress.

Based on this premise, the results of analysis of correlating a measured stress factor with AM-FM based features are reported in this study.

III. Utterances Used in the Study of Stressed Speech

Speech files from two databases, both from the NATO Speech Under Stressed Conditions CD, were used for this study. The first set of files from SUSC-1 contains speech utterances from nine male European (non-English native) fighter controllers that were recorded during communication with fighters. Various stress factors, such as systolic pressure, diastolic pressure, carbon dioxide and heart rate, were measured during the interchange. For the present study, heart rates were chosen to relate with speech features because of the nonavailability of other measurements for some of the speech files in the database. Each speech file in the database was encoded in 16 bits per sample at a rate of 16000 samples per second.

The second speech file analyzed came from an operational aviation emergency, also available in the SUSC-0 CD. This file is discussed in Section V.

Since the database has continuous speech with a large vocabulary of words, it was necessary to choose an utterance that occurred frequently and at varying heart rates for each speaker. Three different heart rates with as large a variation as possible among them were chosen for each speaker. It was found that the utterance "*bull's eye*" most commonly occurred with three different heart rates for many of the speakers.

Using Entropic Waves+ package, the utterance "*bull's eye*" corresponding to each of three different heart rates was extracted for a speaker. From these three files, utterances for the sound "*eye*" were used for correlating AM-FM based features with known heart rates.

IV. Analysis of AM-FM Features with Heart Rates

Inasmuch as the fundamental frequency F0 is commonly observed to rise in stressed speech, modulation features must correlate with the changes in F0 as stress level changes. Table I lists the range of F0 and the formants for speaker RD. Clearly, the difficulty of using F0 alone for detecting heart rate variation is illustrated by this

Table I

Range of fundamental frequency and formants for the utterance "*eye*" for speaker RD at three heart rates

Frequency	At Low Heart Rate (72.27) (RD41)	At Medium Heart Rate (76.49) (RD21)	At High Heart Rate (77.29) (RD55)
F0, Hz	124 – 128	106 - 110	173 – 203
F1, Hz	581 – 779	622 - 773	616 - 711
F2, Hz	1274 – 1632	1282 – 1649	1411 – 1541
F3, Hz	2234 – 2533	2198 – 2490	2421 – 2498
F4, Hz	3331 – 3560	3470 - 3655	3568 - 3607
F5, Hz	3499 – 5051	4652 - 5915	4121 - 5030

table. Although the high heart rate of 77.29 corresponds to high values in the fundamental frequency in speech, medium (76.49) and low (72.27) heart rates do not have proportionally lower values of F0.

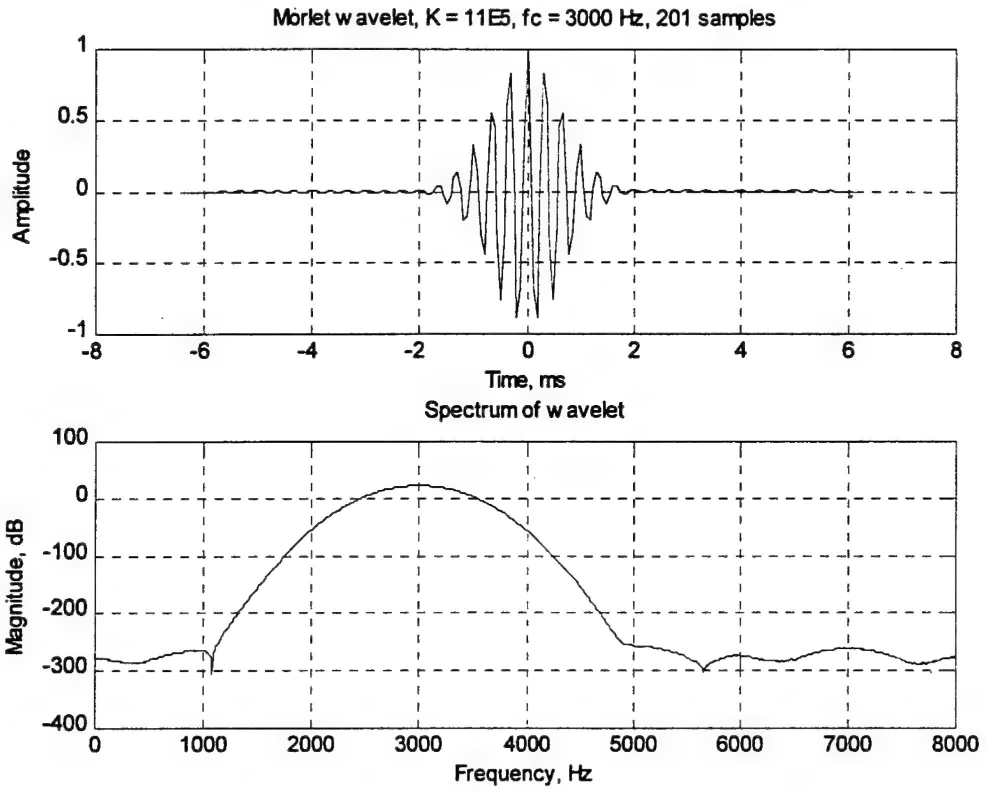


Fig. 1. Wavelet at 3000 Hz and its spectrum

Analysis of AM-FM demodulation features was carried out with narrow bands of speech signals corresponding to the three different heart rates. Narrow band speech signal at center frequency f_c was obtained by bandpass filtering the utterance "*eye*" using the discretized Morlet wavelet function,

$$h(t) = e^{-Kt^2} \cos(2\pi f_c t) \qquad (7)$$

which has an effective RMS bandwidth of

$$\mathrm{BW} \approx \sqrt{\frac{K}{2\pi}} \qquad (8)$$

Typically, a bandwidth of approximately 400 Hz to 600 Hz is chosen for use with the Teager energy and energy separation algorithms. At this bandwidth, demodulation is restricted to the chosen center frequency or formant with negligible effects from neighboring formants [14].

Initially, demodulation was carried out at an arbitrary frequency of 3000 Hz. We note that this frequency falls outside of any formant for the speaker. This choice of arbitrary frequency f_c was used to determine the ability of the modulation model in estimating the fundamental frequency and other features without the need for analysis at estimated formants. Values of $K = 1.1E6$ and $f_c = 3000$ Hz were used in (7) for the bandpass filter wavelet function to get a bandwidth of 418 Hz. Fig. 1 shows the wavelet and its spectrum.

As an example, the speech signal for the utterance "*eye*" at medium heart rate of RD and its bandpass filtered version are shown in Fig. 2 along with their spectra.

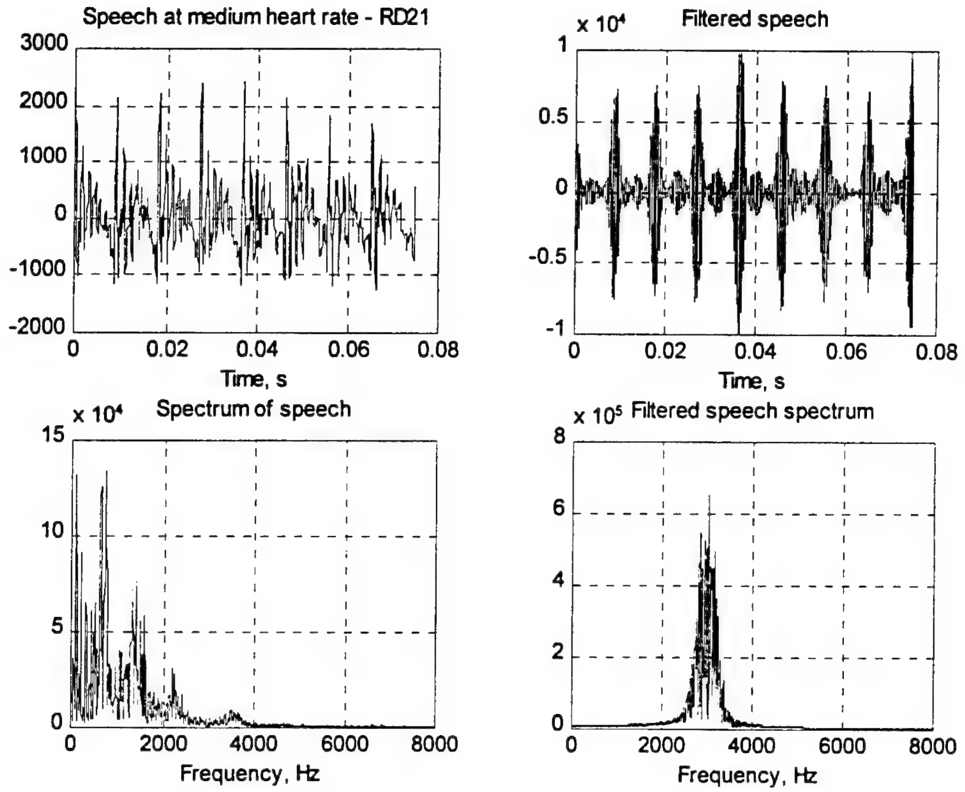


Fig. 2 Original (left), and bandpass filtered (right) speech and their spectra

Using the energy separation algorithms outlined in (5) and (6), the AE and IF for the bandpass filtered signals at low, medium and high heart rates were obtained. These are shown in Fig. 3. The amplitude envelope in each case shows a dominant variation at the rate of the fundamental frequency. This can be seen from the dc-removed spectrum of the AE for each case in Fig. 4. The peaks in the spectra and the harmonics - at 125 Hz, 105 Hz and 180 Hz, respectively, for the low, medium and high heart rates - correspond to the fundamental frequencies estimated using Entropic Xwaves package. Hence, the modulation model provides a method of estimating the fundamental frequency by demodulation using energy separation. The significance of this estimation is that demodulation can be performed at an arbitrary "carrier" frequency (i.e., the center frequency of the bandpass filtered speech). Clearly, demodulation at any convenient frequency eliminates the need for a priori knowledge of the formants for analysis. Since the formants and their bandwidths vary with stress, F0 estimation without formant information helps identify speaker's stress at least qualitatively. The spectrum of IF, however, showed a distinct peak around F0 only at low or medium stress level.

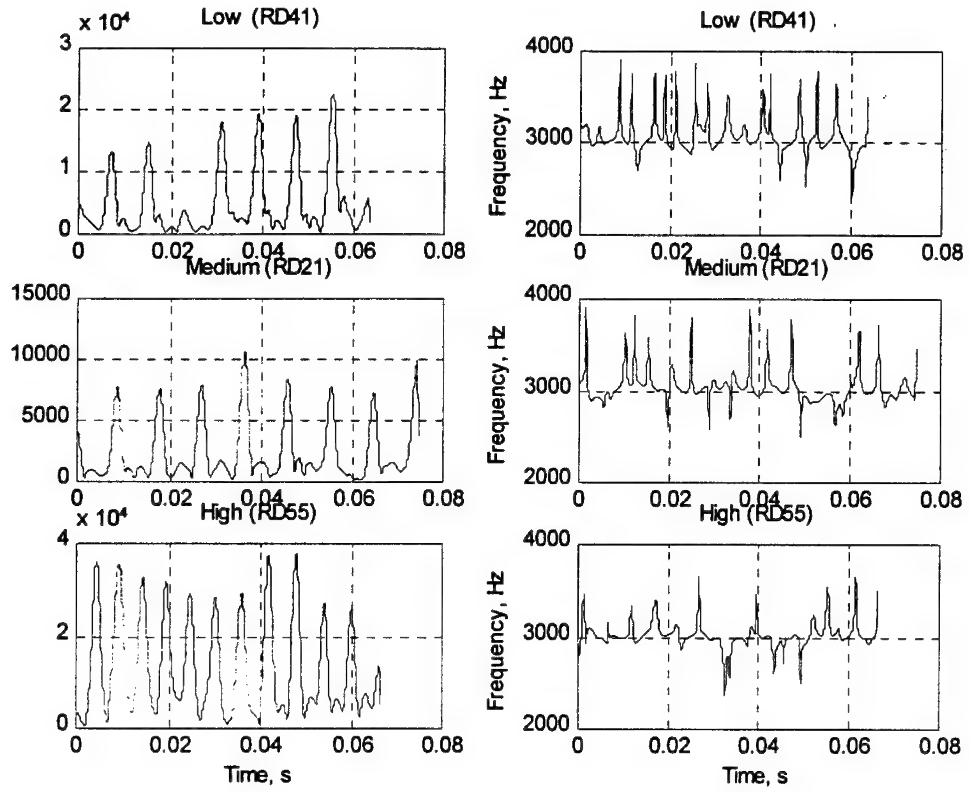


Fig. 3 Amplitude envelope (left), and instantaneous frequency (right), around 3000 Hz for speaker RD at low, medium and high heart rates

In addition to a higher estimation of F0 at high heart rate, the spectrum of AE also shows fewer, but blended peaks indicating a multipulse glottal excitation with at a single dominant rate. Thus high heart rates may be characterized by merged spectral peaks at higher than nominal frequencies in AE.

An attempt was made to model the amplitude envelope as arising from a secondary linear system. In this model the amplitude envelope was assumed to be the output of a linear prediction (LP) system, analogous to the vocal tract model. Since AE, the output of the system, is quasiperiodic arising from the demodulation of a voiced segment of speech, the LP spectrum may represent a secondary system whose characteristics vary with stress. A high order (≥ 7) LP model was derived for the AE. Fig. 5 shows the LP spectra for speaker RD at three heart rates. Each spectrum was obtained from the model

$$H_s(z) = \frac{G}{1+\sum_{k=1}^{p} a_k z^{-k}} \qquad (9)$$

where $H_s(z)$ is the secondary system with output AE. As seen from Fig. 5, at low and medium heart rates, the LP spectrum is a bandpass type with Q factor (the ratio of center frequency to 3-dB bandwidth) less than unity. At high heart rate, the spectrum becomes narrow with Q > 1.

Analysis at a Formant: Although the previous analysis at an arbitrary frequency gives a good estimate of F0, the original AM-FM model (1) proposed by Maragos, Quatieri, and Kaiser [5] was used to analyze nonlinearities at each resonance. From a physical point of view, analysis at a formant describes the time-varying and nonlinear phenomena around the resonant frequency of the vocal tract filter. Due to stress, the vocal tract characteristics may vary leading to changes in the formants and hence the modulations around them. To assess these variations with heart rates, demodulation feature analysis around formants was carried out.

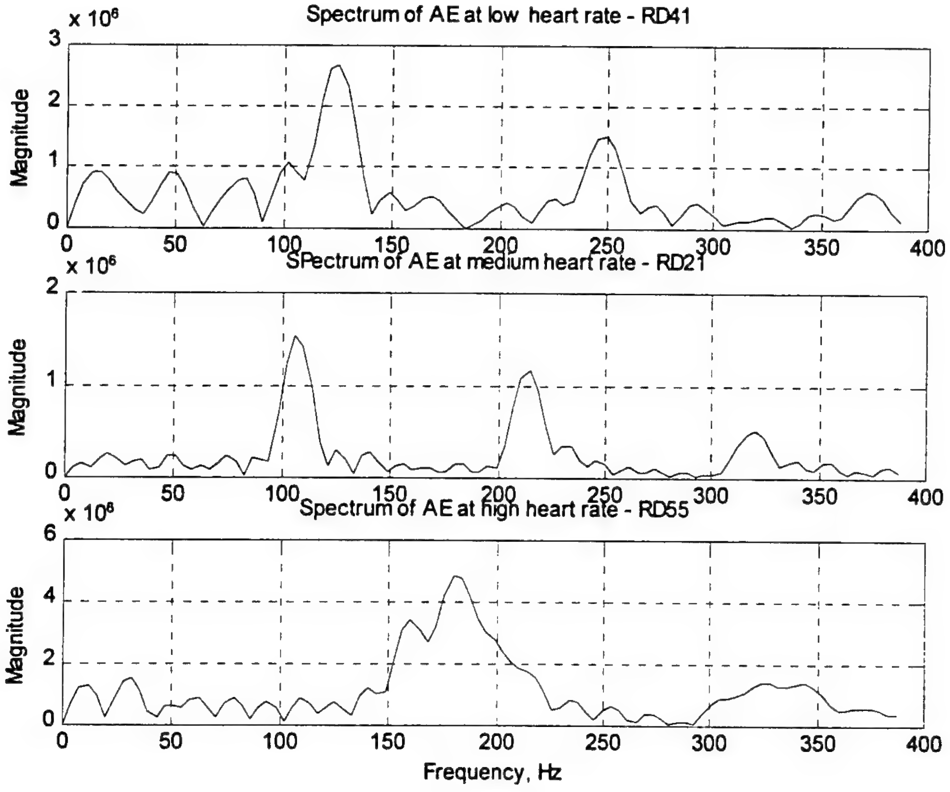


Fig. 4 Spectra of amplitude envelopes at different heart rates for speaker RD

It has been observed [15] that the vowel spectrum is modified at high frequencies in stressful situations. This modification alters the formant frequencies and the energy in higher frequency bands. Because of these changes at high frequencies with stress, demodulation characteristics for the vowel utterance at formants F3 and F4 were studied. The analysis frequencies of $f_c = 2300$ Hz and 3500 Hz were chosen from the formants of the "neutral" utterance at medium heart rate for speaker RD.

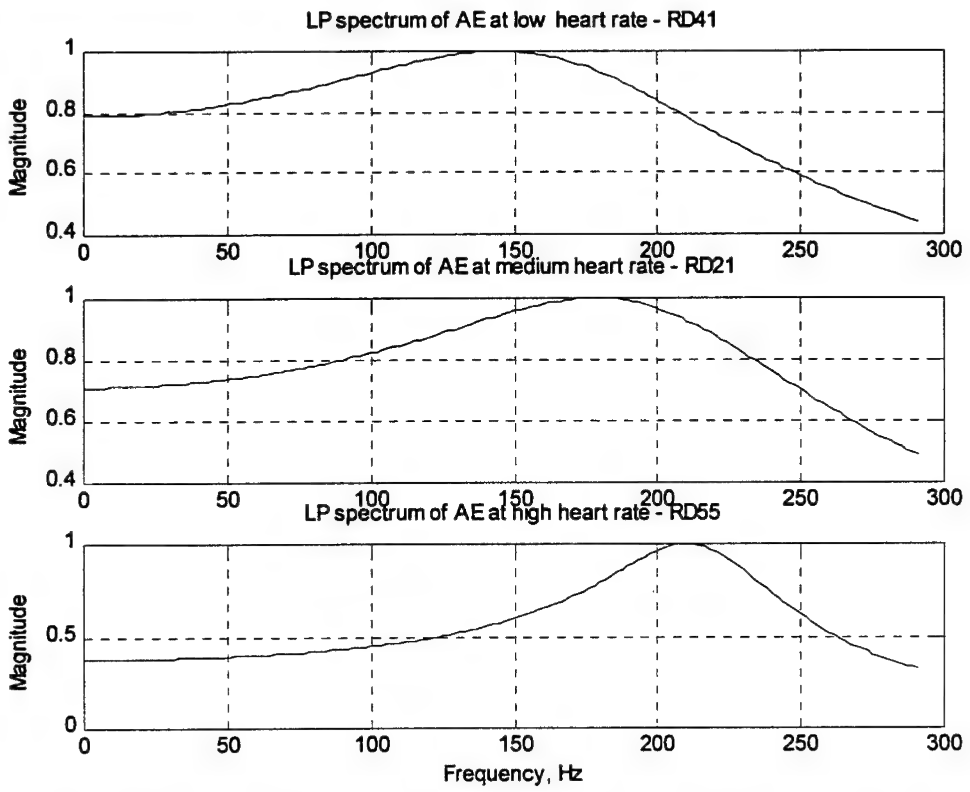


Fig. 5 Normalized linear prediction spectra of amplitude envelopes at different heart rates for speaker RD.

Analysis at f_c = F3 = 2300 Hz: Fig. 6 shows the AE and IF for the utterance "*eye*" after bandpass filtering using a narrow band of 282 Hz at $f_c = 2300$ Hz for the three heart rates of RD. As seen in this figure, the instantaneous frequency variation appears to be slightly more for the utterance at high heart rate than for the other two. The mean and standard deviation of IF do not show significant variation among the three cases. The spectrum of the AE for each case, given in Fig. 7, shows a distinct peak at the fundamental frequency. As with the demodulation analysis at 3000 Hz, the peaks around F0 are dominant (at 125 Hz and 105 Hz) with much smaller peaks at its harmonics for the low and medium heart rates; the peak for the high heart rate case is blended (between 160 Hz and 180 Hz) with indistinct harmonic peaks. Thus the utterance at high heart rate appears to generate closely spaced multipulse excitation.

The LP spectra for the three cases are shown in Fig. 8. As with the case of analysis at the arbitrary frequency of 3000 Hz, the sharp bandpass spectra have center frequencies in the neighborhood of their corresponding F0 values. Table II shows the center frequencies, bandwidths and Q factors for each heart rate. For comparison, the fundamental frequencies given by Entropic Xwaves package and those estimated from the spectra of AE are also given in the table. As seen from the table, the center frequencies of the bandpass LP spectra appear to be closer to the F0 values given by the Xwaves package. However, Q values of LP spectra of AE do not correlate with heart rate.

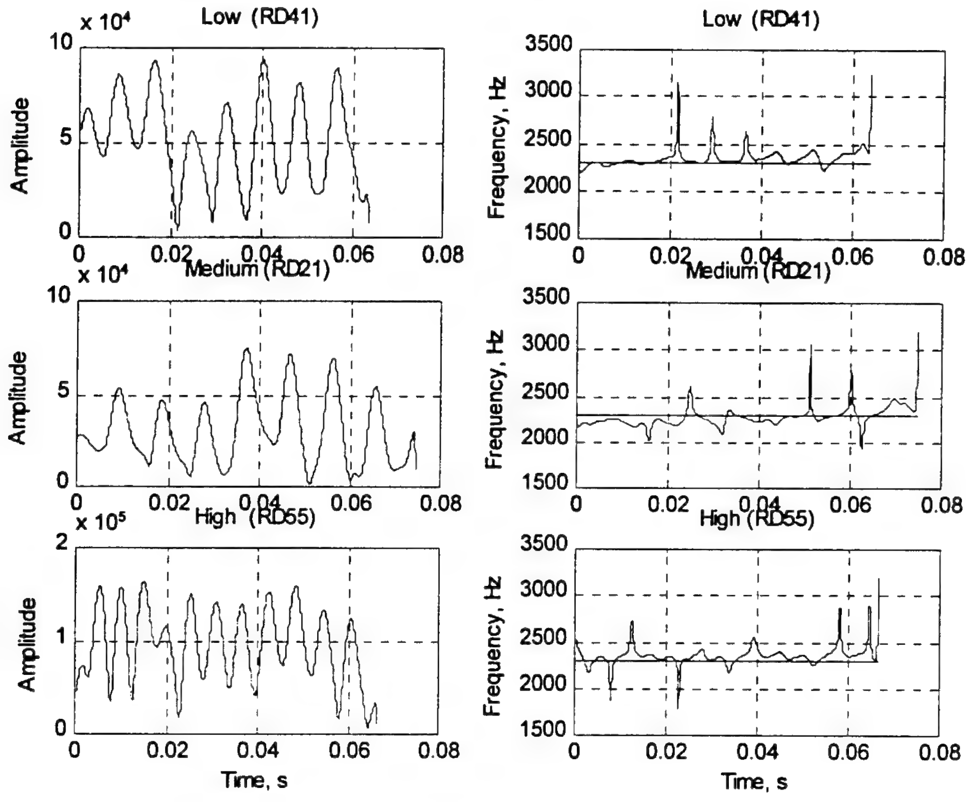


Fig. 6 Amplitude envelope (left), and instantaneous frequency (right), around F3 = 2300 Hz for speaker RD at low, medium and high heart rates. The analysis frequency is shown by the horizontal line in IF.

Table II

Peak frequencies and bandwidth of LP spectra of AE at f_c = F3 = 2300 Hz

Heart rate	Center frequency, Hz	3-dB bandwidth, Hz	Q	F0 from AE spectrum, Hz	F0 range from Xwaves, Hz
Low (RD41)	116	35	3.31	125	124 – 128
Medium (RD21)	106	23	4.61	105	106 - 110
High (RD55)	162	55	2.95	180	173 – 203

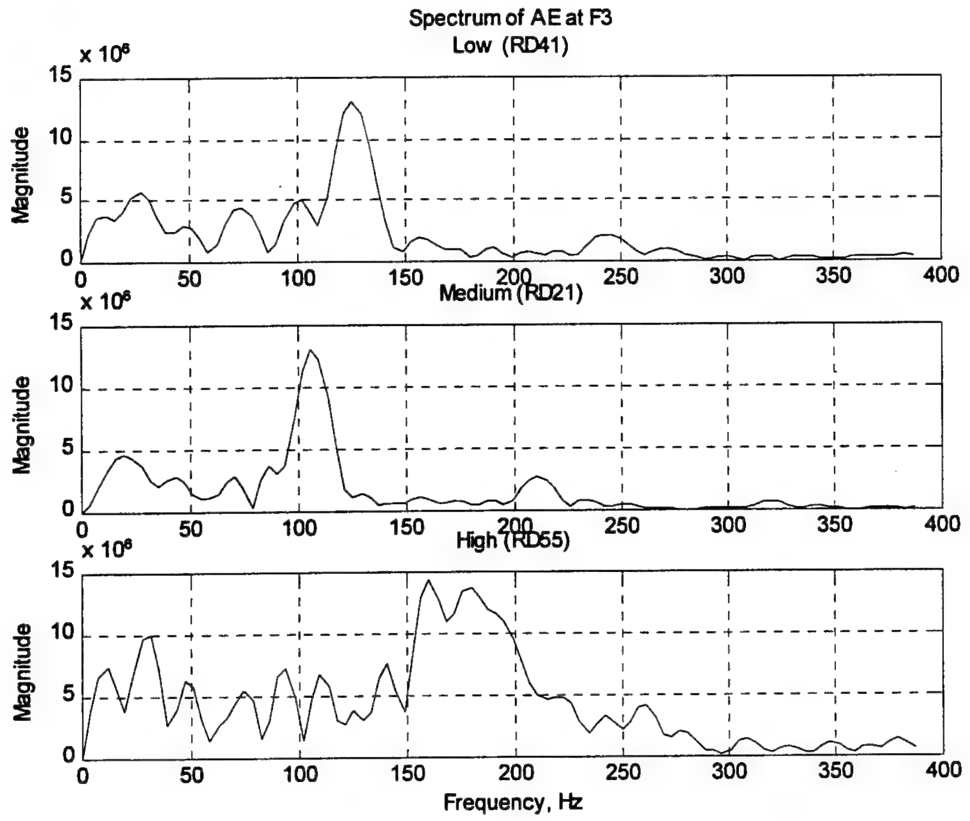


Fig. 7 Spectra of amplitude envelopes at the analysis frequency of F3 = 2300 Hz for utterances at different heart rates for speaker RD

Demodulation in the vicinity of F3 can be used to estimate the formant and its bandwidth. From the instantaneous frequency f_i, the unweighted estimates of the formant (F_u) and its bandwidth (B_u) are given by [14, 16]

$$F_u = \overline{f_i} = \frac{1}{T}\int_{t_0}^{t_0+T} f_i(t)dt \tag{10}$$

$$[B_u]^2 = \overline{(f_i - F_u)^2} \tag{11}$$

Using the first and second weighted moments with mean squared value of the AE as weight, the weighted estimates of the formant (F_w) and its bandwidth (B_w) are given by [14, 16]

$$F_w = \frac{\overline{f_i[a(t)]^2}}{\overline{[a(t)]^2}} = \frac{\left(\frac{1}{T}\right)\int_{t_0}^{t_0+t} f_i(t)[a(t)]^2\, dt}{msq(a)} \tag{12}$$

Bandwidth can be split into contributions from FM and AM. Bandwidth due to variations in IF is given by

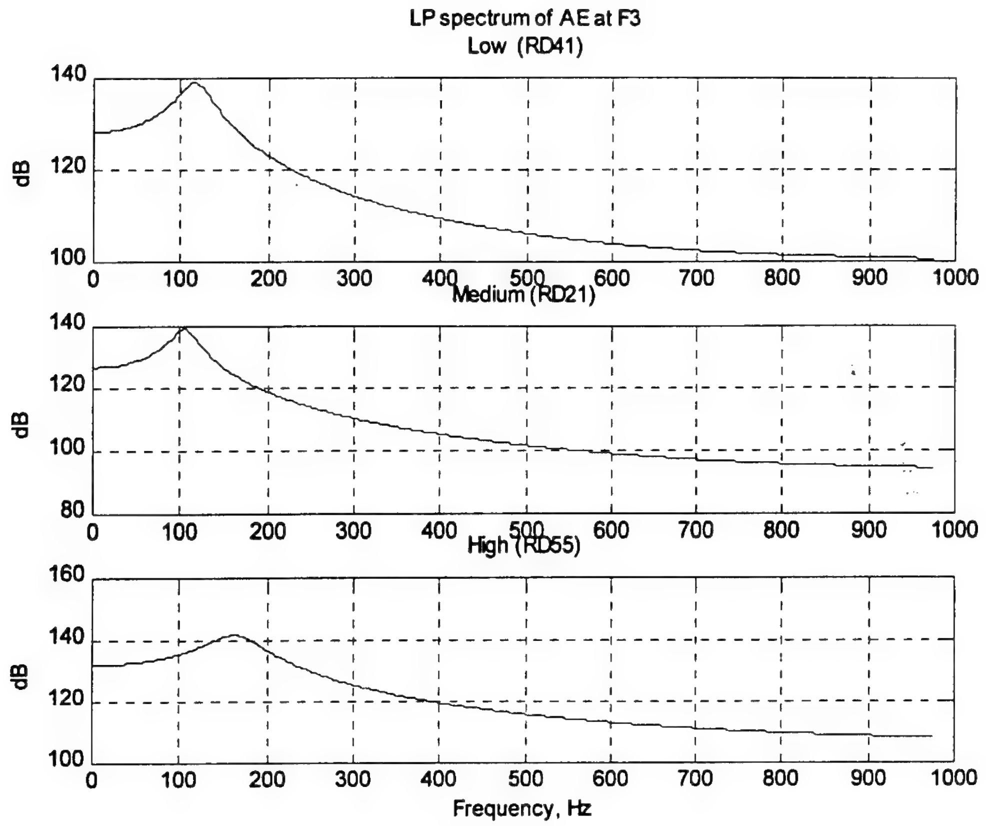


Fig. 8 LP spectra of AE at different heart rates for speaker RD at F3 = 2300 Hz.

$$[B_{wf}]^2 = \frac{\overline{[f_i - F_w]^2[a(t)]^2}}{\overline{[a(t)]^2}} = \frac{\left(\frac{1}{T}\right)\int_{t_0}^{t_0+T}[f - F_w]^2[a(t)]^2\,dt}{msq(a)} \tag{13}$$

while the AM contribution to bandwidth is given by

$$[B_{wa}]^2 = \frac{1}{4\pi^2}\frac{\overline{[a'(t)]^2}}{\overline{[a(t)]^2}} = \frac{1}{4\pi^2}\frac{\frac{1}{T}\int_{t_0}^{t_0+T}[a'(t)]^2\,dt}{msq(a)} \tag{14}$$

where a(t) is the amplitude envelope of the bandpass filtered signal $s(t) = a(t)\cos\Phi(t)$, and

$$msq(a) = \overline{[a(t)]^2} = \frac{1}{T}\int_{t_0}^{t_0+T}[a(t)]^2\,dt \tag{15}$$

The total weighted bandwidth is then obtained from

$$[B_w]^2 = [B_{wf}]^2 + [B_{wa}]^2 \tag{16}$$

Since stress contributes to changes in the formants and their bandwidths, analysis at F3 also included the unweighted and weighted formants and bandwidths. These values are shown in Table III. The weighted and unweighted formants and the bandwidths appear to show a decrease in values with increased heart rate. These changes are consistent with changes in vocal tract resonant frequencies due to stress observed by Ruiz and Legros [15], i.e., stress may increase or decrease the resonant frequencies and their bandwidths. The bandwidth due to amplitude modulation, though small, shows a slight increase with heart rate.

Table III

Unweighted and weighted formants and bandwidths at demodulation analysis frequency of F3 = 2300 Hz

Heart rate	F_u, Hz	B_u, Hz	F_w, Hz	B_{wf}, Hz	B_{wa}, Hz	B_w, Hz
Low (RD41)	2346	96	2319	42	0.003	42
Medium (RD21)	2276	109	2269	43	0.0038	43
High (RD55)	2350	106	2337	40	0.0039	40

Analysis at f_c = F4 = 3500 Hz: At the higher formant of F4, the spectra of AE for the three heart rates show peaks at their corresponding values of F0 (Table IV), as with the analyses at F3 and at 3000 Hz. As seen in Fig. 9, the AE varies at the fundamental frequency F0 and the IF shows large variations about the analysis frequency F4, both similar to variations at f_c = F3 = 2300 Hz shown in Fig. 6. The center frequencies and the 3-dB bandwidths of the LP spectra of AE also show similar, uncorrelated variation with heart rates. While the unweighted formants (Table V) appear to be unrelated, the unweighted formant bandwidths are seen to decrease with increasing heart rates. Similarly, the weighted formants show an increase with increasing heart rates. Similar results were obtained for analysis at the arbitrary frequency of 3000 Hz.

Table IV

Peak frequencies and bandwidth of LP spectra of AE at f_c = F4 = 3500 Hz

Heart rate	Center frequency, Hz	3-dB bandwidth, Hz	Q	F0 from AE spectrum, Hz	F0 range from Xwaves, Hz
Low (RD41)	123	50	2.46	121	124 – 128
Medium (RD21)	129	85	1.52	106	106 - 110
High (RD55)	173	40	4.33	175	173 – 203

Table V

Unweighted and weighted formants and bandwidths at demodulation analysis frequency of F3 = 3500 Hz

Heart rate	F_u, Hz	B_u, Hz	F_w, Hz	B_{wf}, Hz	B_{wa}, Hz	B_w, Hz
Low (RD41)	3519	142	3479	23	0.0053	23
Medium (RD21)	3507	83	3509	45	0.0049	45
High (RD55)	3555	76	3536	39	0.0043	39

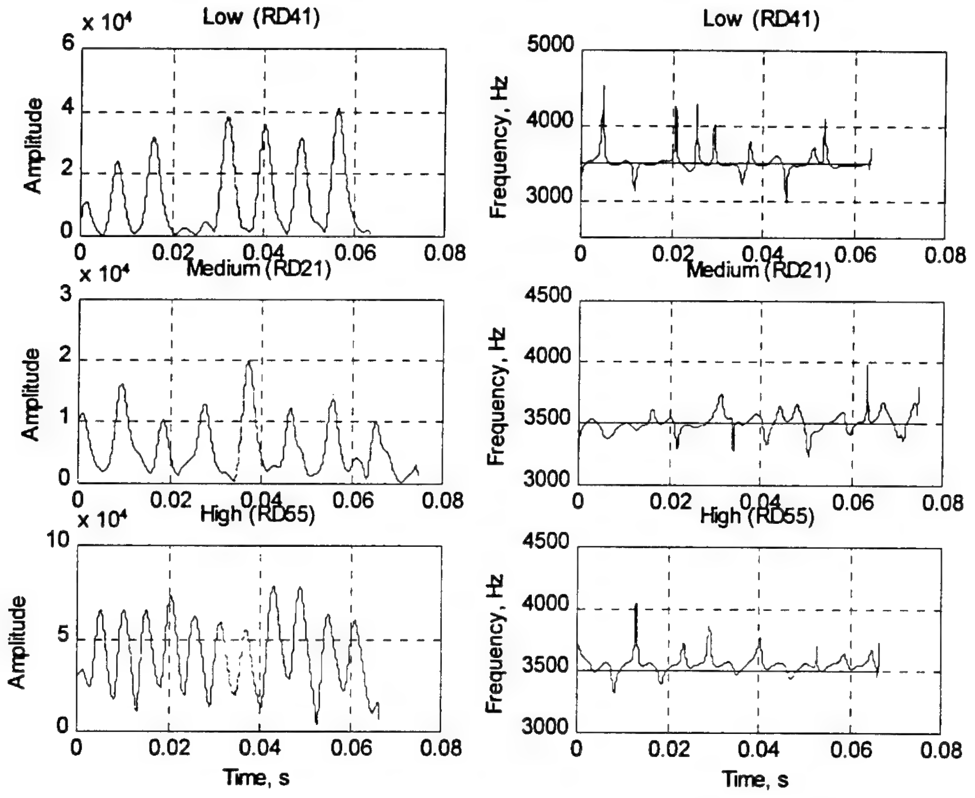


Fig. 9 Amplitude envelope (a), and instantaneous frequency (b), around F3 = 3500 Hz for speaker RD at low, medium and high heart rates. The analysis frequency is shown by the horizontal line in IF.

Tracking of formants and bandwidths in each frame: Instead of analyzing the entire utterance "*eye*" as a single frame containing about eight pitch intervals, demodulation parameters were tracked by segmenting the utterance into frames of approximately pitch duration. This tracking provides variation of F0 and formant estimates and other features with time. Using frame lengths of 15 ms, utterance at each of the three heart rates was analyzed every 5 ms. Variation of F0 estimates from the AE spectral peaks at f_c = F3 = 2300 Hz for each heart rate is shown in Fig. 10. In this figure, the utterance at high heart rate is clearly distinguishable from those at low and medium rates. As for the lower values of F0 at medium heart rate compared to those at low, it appears that a deviation in heart rate – up or down – from "neutral" contributes to an increase in F0. Profiles of peak frequencies in the LP spectra of AE, given in Fig. 11, also show almost identical variations with time as the F0 estimate profile in Fig. 10. The unweighted and weighted formants shown in Fig. 12, indicate higher variations about the analysis frequency of 2300 Hz for utterances at heart rate below and above the "neutral" rate. This agrees with the higher range of formant (F3) values at low and high heart rates given in Table I. Thus the estimate of F3 by demodulation around "neutral" F3 is able to track the time variation. The trajectory of F3 estimate, however, is unable to differentiate between lower and higher heart rates relative to medium rate.

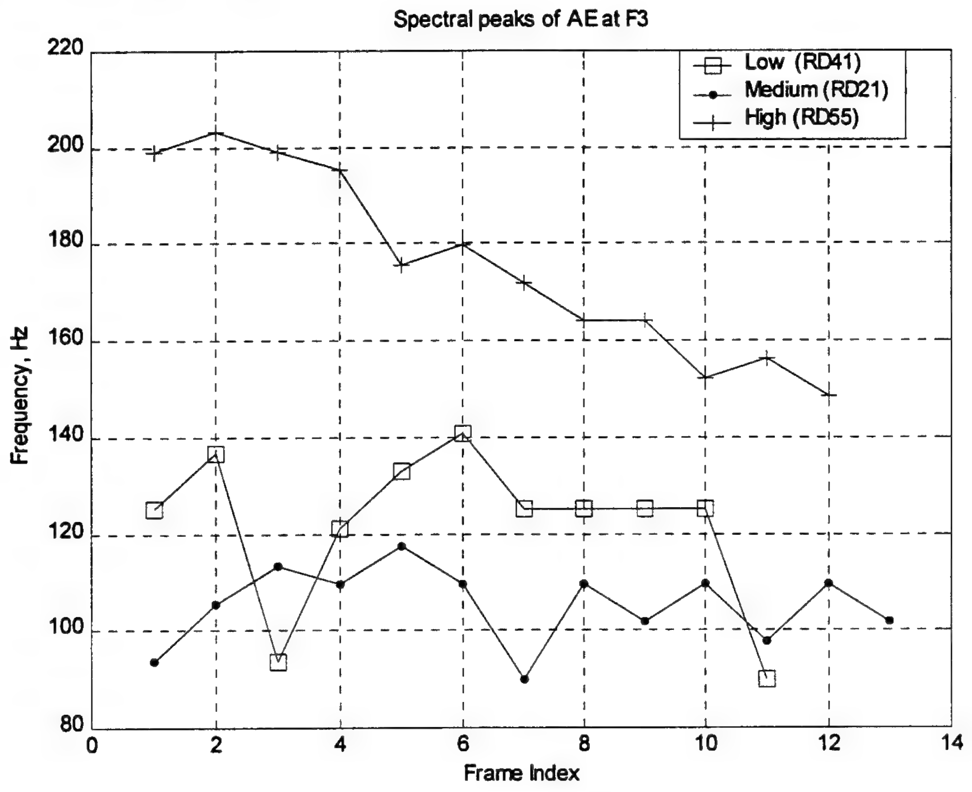


Fig. 10 Tracking of F0 estimate from AE spectral peaks with frames at f_c = F3 = 2300 Hz.

Analysis around f_c = nominal F4 = 3500 Hz shows higher F0 estimates and higher frequencies in the LP spectrum of AE for high heart rate as seen in Fig. 13. The unweighted formant profiles shown in Fig. 14, however, indicate generally higher F4 values at high heart rate while the low and medium rates are indistinguishable. This is in contrast to the Xwave formants (Table I) which show increasing range of F4 with increasing heart rate. The weighted formant, on the other hand, appears to correlate somewhat linearly with heart rate.

The tracking analyses at F3 and F4, in general, show that the mean and median values of F0 estimates over all the frames – based on AE spectral peaks and LP spectral peaks of AE - are higher at low and high heart rates than those at neutral heart rate.

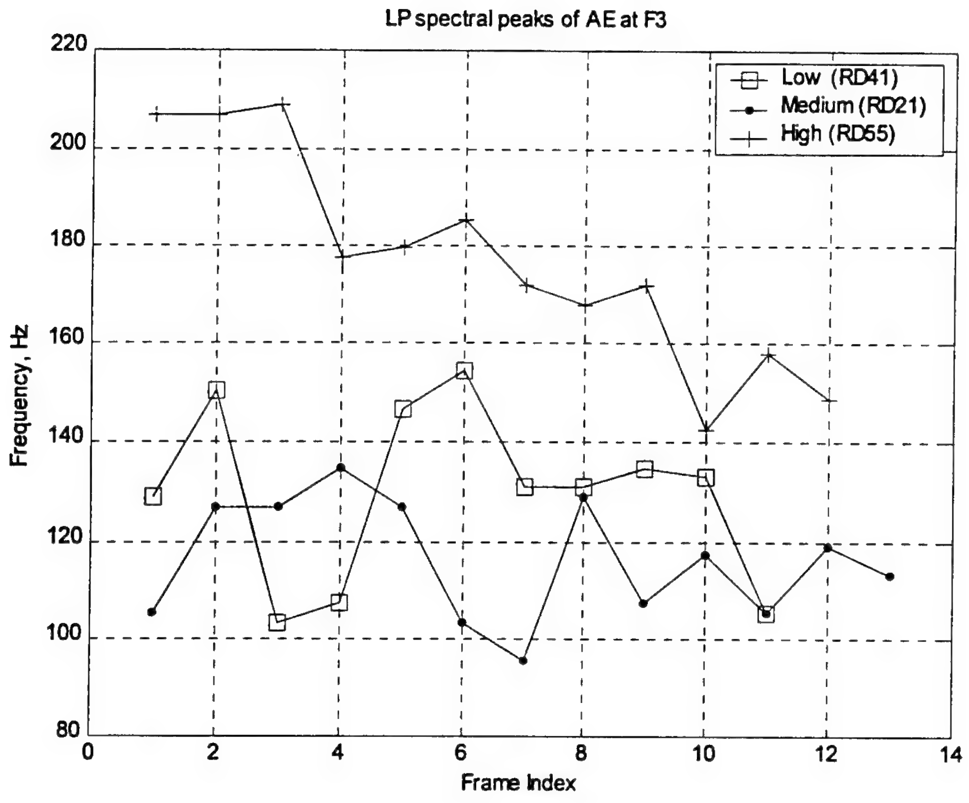


Fig. 11 Tracking of LP spectral peaks of AE with frames at f_c = F3 = 2300 Hz.

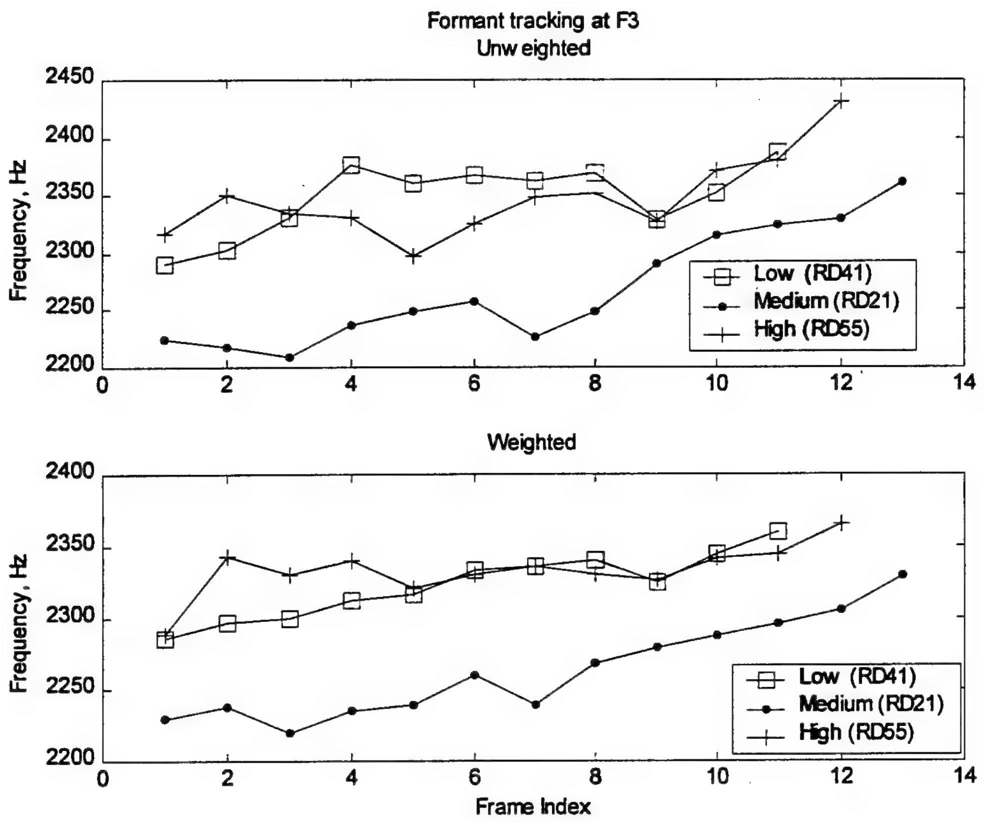


Fig. 12 Unweighted and weighted formant trajectory at f_c = F3 = 2300 Hz

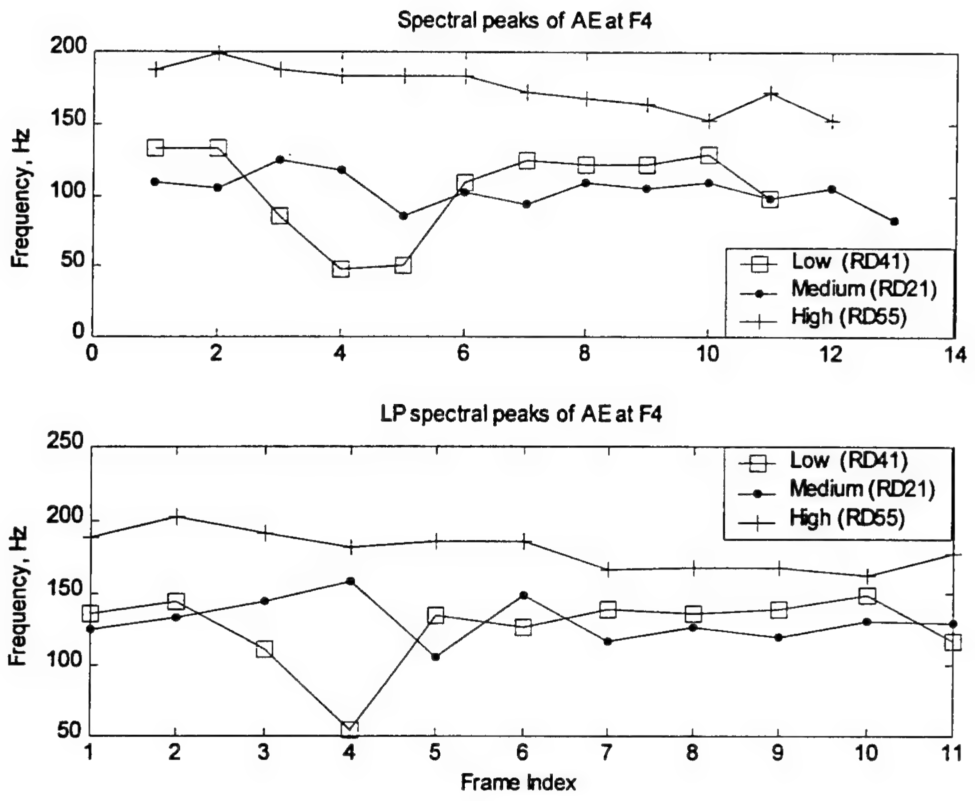


Fig. 13 Tracking of peaks of AE spectrum and AE LP spectrum with frames at f_c = F4 = 3500 Hz

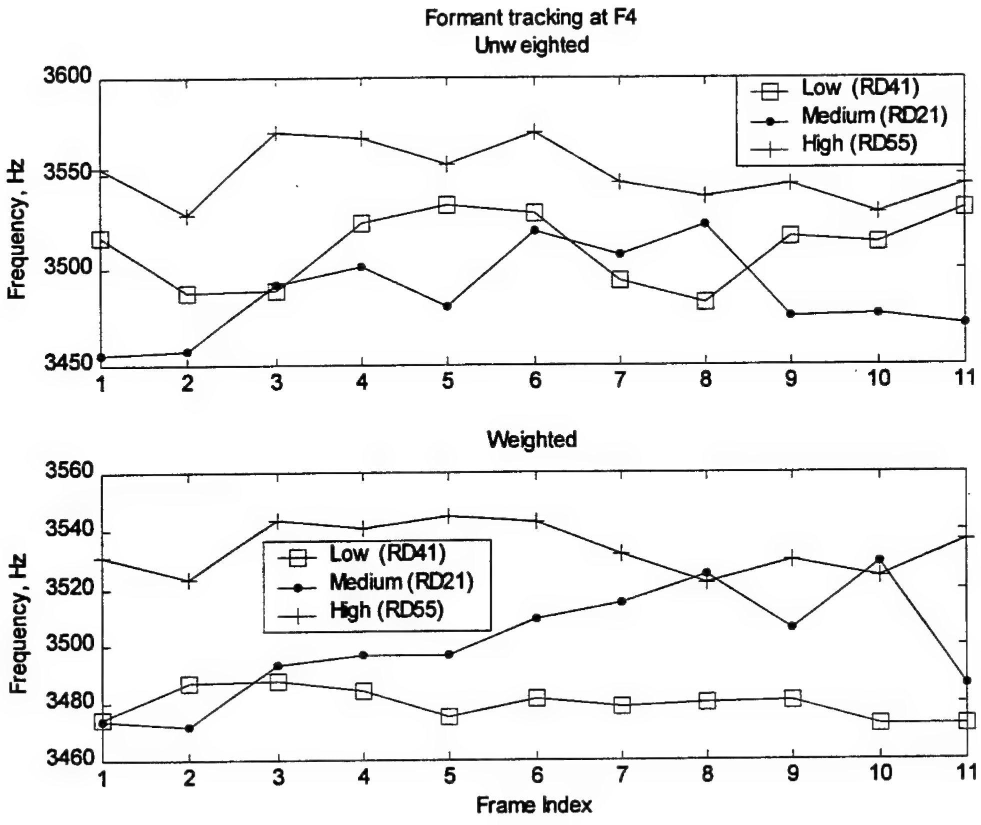


Fig. 14 Tracking of unweighted and weighted formants with frames at f_c = F4 = 3500 Hz

Analysis at lower frequencies: Performance of the energy separation algorithm is poor at low frequencies. Moreover, Morlet wavelet bandpass filter requires a large size for reasonable sharpness in the passband. Hence, demodulation below F3 was not carried out.

V. Feature Analysis of Speech from an Operational Aviation Emergency

The NATO database SUSC-0 was used to study the AM-FM characteristics of speech in an emergency. This database consists of approximately 10 minutes of interchange between an F-16 pilot, his wingman, and the tower in an in-flight emergency. During the course of flight, the pilot notices the hydraulic oil pressure dropping, realizes the loss of his engine, declares 'mayday' emergency, and, with the help of his wingman and the tower, lands the aircraft safely. Although the pilot's voice during the course of emergency was remarkably calm, one can assess the stress in this life-threatening situation.

For analysis, the utterance */is/* by the pilot was extracted from his speech during (a) his checking of the instrument panel, (b) his noticing of oil pressure being low, (c) his declaration of emergency, (d) his declaration of 'mayday', and (e) his urgent request for help in landing. The context for the utterance */is/* with its approximate time marking in the database is shown in Table VI for each of the five cases to indicate the level of stress. The above utterance segments were concatenated for analysis to track the modulation parameters as the stress level varied from neutral (vi3-is-25), to high (vi3-is-415 and vi3-is-440) and back to medium or low level (vi3-is-450). Fig. 15 shows the collection of all five utterances.

Table VI

Utterance context for pilot viper3

No.	File name	Utterance context	Approximate time marker in the database	Number of samples
1	vi3-is-25	oil pressure *is* about ..	25 s	7781
2	vi3-is-249	... know what an sfo *is* ...	249 s	3322
3	vi3-is-415	... emergency zero four *is* ...	415 s	1780
4	vi3-is-440	... it *is* ... lost my engine ... mayday mayday mayday	440 s	3327
5	vi3-is-450	... *is* an emergency I'm engine out	450	2125

Modulation analysis was carried out at the center frequency of 3300 Hz, which is close to the third formant F3. Since utterances at different stress levels were combined together, modulation features were tracked every 5 ms using 15 ms frames. The unweighted and weighted formant trajectories are shown in Fig. 15. These smoothed F3 profiles show that as stress increases formant F3 also increases. Similar increases were also observed in the profiles of median and mean-squared values of the instantaneous frequencies with frames. The amplitude envelope variations with frames, on the other hand, did not show similar correspondence with stress. Fig. 16 gives the profile of the peak frequencies in the spectra of the amplitude envelopes, which are the estimates of F0, with frames. This profile shows large variations in F0 at high and medium stress levels relative to F0 corresponding to neutral utterance.

Analysis at 1400 Hz, which is in the vicinity of F2, showed similar profiles of formants and F0 estimates.

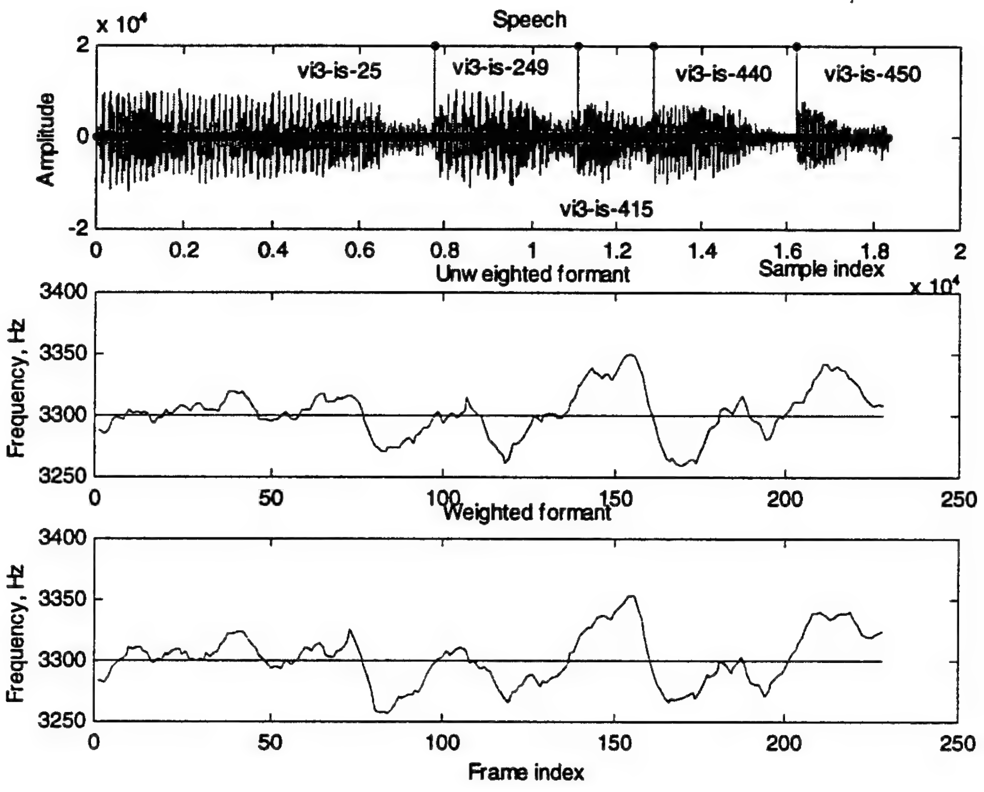


Fig. 15 Concatenated speech (top), unweighted formant (middle) and weighted formant profiles ifor speaker vi3 at analysis frequency 3300 Hz. (The formant profiles are smoothed.) Vertical lines in the speech graph separate utterances at different stress levels.

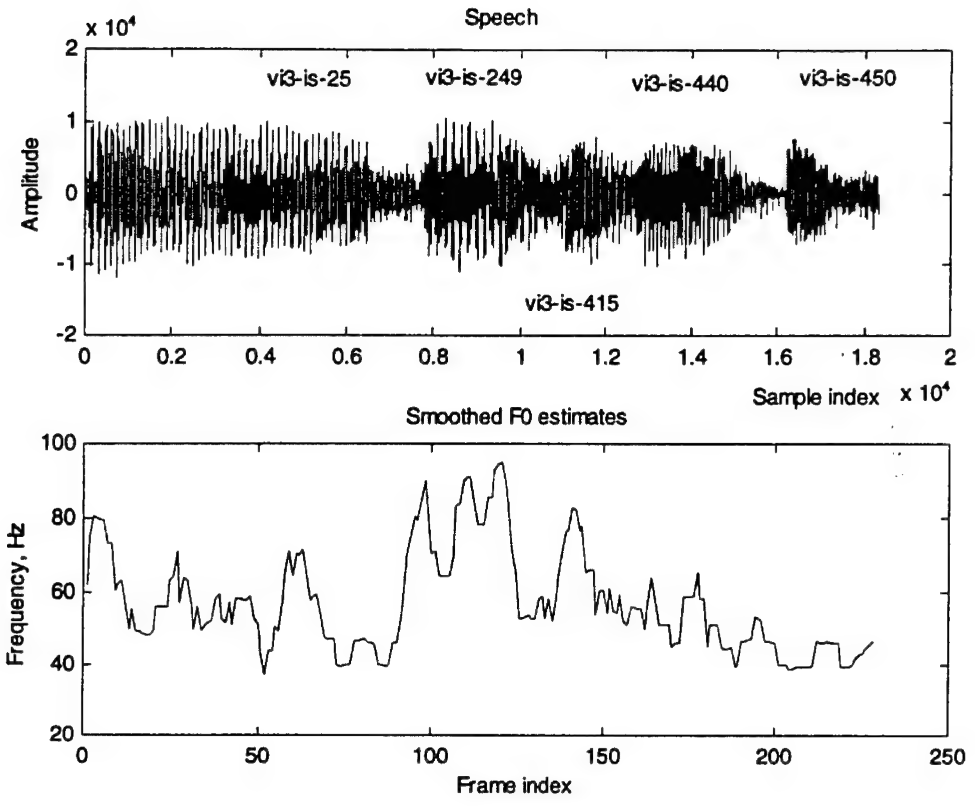


Fig. 16 Speech signal and smoothed F0 estimates at analysis frequency 3300 Hz

VI. Discussion

Two sets of utterances, one spoken at three different heart rates and the other at different levels of stress in an emergency, have been analyzed using the AM-FM model. Results of these analyses show that the modulation based parameters provide stress related information comparable to that from the direct analysis of speech for vocal tract and other parameters. The fundamental frequency of voicing, long observed as related to stress, is clearly brought out by the spectrum of the amplitude envelope in the modulation model. Since speech signal amplitude is generally observed to have a large value at the onset of excitation and decay before the next excitation, the amplitude envelope may be modeled as $a_k(t) = e^{-\sigma_k t} A_k(t)$. The exponential decaying factor in this model may provide additional information related to stress. The decaying amplitude model clearly requires pitch synchronous analysis and tracking. Since the AM-FM model provides an estimate of F0 at any arbitrary analysis frequency, it facilitates pitch synchronous analysis.

Instantaneous frequency from the AM-FM model is able to lock in with the nearest formant when the analysis is carried out in the vicinity of an estimated formant. Since speech signal energy at high frequencies is observed to increase with stress, estimation of higher formants and their bandwidths is a useful tool for stressed speech analysis.

VII. Conclusion

Analyses of two sets of utterances showed that the AM-FM model brought out stress related variations of F0 and formants. Modeling of the amplitude envelope as arising from a secondary linear system activated by stress did not correlate with stress; this modeling, however, resulted in another estimate of F0. Additionally, the modulation model showed that stress related F0 can be estimated by demodulation around any arbitrary frequency without first finding a formant. In general, the modulation model indicated higher values of F0 whenever the speaker's heart rate was different from the neutral rate; also, the formant values from the model showed an increase with actual stress. Both of these results are consistent with F0 and formant variations observed by direct analysis of speech under stress. More data sets of speech under actual stress need to be analyzed for better correlation of F0 and formants with stress.

References

1. Teager, H.M., and S. Teager, "Evidence for Nonlinear Production Mechanisms in the Vocal Tract," NATO Advanced Study Inst. On Speech Production and Speech Modeling, Bonas, France, 1989, Kluwer Acad. Pub., 1990.
2. H.M. Teager, "Some Observations on Oral Air Flow during Phonation," IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-28, No. 5, pp 599-601, Oct. 1980.
3. T.F. Quatieri, T.E. Hanna and G.C. O'Leary, "AM-FM Separation using Auditory-Motivated Filters," IEEE Trans. Speech and Audio Proc., Vol. 5, No. 5, pp. 465-480, Sep. 1997.
4. A.B. Fineberg, R.J. Mammone and J.L. Flanagan, "Application of the Modulation Model to Speech Recognition," Proc. ICASSP '92, pp. I-541-I-544, 1992.
5. P. Maragos, T.F. Quatieri, and J.F. Kaiser, "Speech Nonlinearities, Modulations, and Energy Operators," Proc. ICASSP '91, pp. 421-424, 1991.
6. C.E. Williams, and K.N. Stevens, "Vocal Correlates of Emotional Stress," in Speech Evaluation Psychiatry, J.K Darby, Jr. (Ed.), Grune & Stratton, Inc.,1981.
7. P. Maragos, J.F. Kaiser and T.F. Quatieri, "On Amplitude and Frequency Demodulations using Energy Operators," IEEE Trans. Signal Processing, Vol. 41, No. 4, pp. 1532-1550, Apr. 1993.
8. J.F. Kaiser, "On a Simple Algorithm to Calculate the 'Energy' of a Signal, Proc. ICASSP '90, pp. 381-384, 1990.
9. K. Gopalan, "Amplitude and Frequency Modulation Characteristics of Stressed Speech," Final Report, AFOSR Summer Faculty Research Program, Bolling AFB, July 1998.

10. Lieberman, P. and S.B. Michaels, "Some Aspects of Fundamental Frequency and Envelope Amplitude as related to the Emotional Content of Speech," Acoust. Soc. Am. Vol. 34, No. 7, 1962, pp. 922-927.
11. E. Absil, et al., "Time-related Variabilities in Stressed Speech under Laboratory and Real Conditions," Technical Proceedings, Workshop on Speech under Stress Conditions, NATO Defence Research Group, Sep. 1996, pp. 23-1 - 23-4.
12. P. Benson, "Analysis of the Acoustic Correlates of Stress from an Operational Aviation Emergency," *ibid*, pp. 25-1 - 25-4.
13. A. Protopapas, and P. Lieberman, "Effects of Vocal F0 Manipulations on Perceived Emotional Stress," *ibid*, pp. 4-1 - 4-4.
14. A. Potamianos, "Speech Processing Applications using an AM-FM Modulation Model," Ph.D. Dissertation, Harvard University, 1995.
15. R. Ruiz and C. Legros, "Vowel Spectral Characteristics to Detect Vocal Stress," Proc. 15th Int. Congress on Acoustics, Trondheim, Norway, pp. 141-144, June 1995.
16. L. Cohen and C. Lee, "Instantaneous bandwidth," in *Time-Frequency Signal Analysis – Methods and Applications* (B. Boashash, Ed.), London: Longman-Chesire, 1992.

A STUDY ON ACCELERATING THE RAY/BÉZIER-PATCH INTERSECTION COMPUTATION IN XPATCH

Donald L. Hung
School of Electrical Engineering and Computer Science

Washington State University, Richland
2710 University Drive
Richland, Washington 99352

Final Report for:
Summer Research Extension Program
Information Directorate (IFSD)
Wright Research Site
Dayton, Ohio

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Wright Research Site
Dayton, Ohio

December 1999

A STUDY ON ACCELERATING THE RAY/BÉZIER-PATCH INTERSECTION COMPUTATION IN XPATCH

Donald L. Hung
School of Electrical Engineering and Computer Science
Washington State University, Richland

Abstract

Xpatch is a software program which uses the ray-tracing technique to model the incident radar signal and generate radar cross sections of various objects. Surfaces of the objects are modeled by shape primitives such as boxes, triangular facets and bicubic Bézier patches. The major computation associated with the ray-tracing technique is to determine whether a given ray intersects with the shape primitives. In general, ray-tracing algorithms (with respect to different shape primitives) are computationally intensive, which is particularly true for ray/Bézier-patch intersection algorithms owing to their higher degree of complexity. In this study, we focus on accelerating the ray/Bézier-patch intersection computation because for object modeling, Bézier patches allow much higher accuracy comparing with other shape primitives. As the outcomes of the study, a new algorithm suitable for physical implementation has been developed; its corresponding architecture for physical realization has also been proposed. In addition, performance estimation has been obtained based on implementation models relying on digital signal processors (DSPs). This information can be used as references for future works on practical implementation.

A STUDY ON ACCELERATING THE RAY/BÉZIER-PATCH INTERSECTION COMPUTATION IN XPATCH

Donald L. Hung

1. Introduction

Ray tracing is one of the most important rendering techniques in the field of computer graphics for producing high-quality and photorealistic pictures. When rendering a scene, at least one ray is cast backward for each pixel of the picture. Each cast ray needs to be determined whether it intersects the given scene. If it hits the object, it may produce other rays, such as reflection ray and transparent ray. The ray tracing repeats this fashion so on and so forth until each pixel gets its color value. In order to avoid aliasing effects, sometimes it is necessary to cast multiple rays for each pixel. Therefore the ray-tracing technique deals with hundreds of and thousands of ray-object-intersections and is well know for its exorbitant computation.

Xpatch is a software program that uses the ray-tracing technique to model the incident radar signal and generate radar cross-section predictions of various objects. The object surfaces are modeled by different shape primitives such as boxes, triangular facets, bicubic Bézier patches, etc.. Corresponding to each type of the shape primitives, a certain ray-tracing algorithm is executed in order to determine if a given ray intersects with the interested shape primitives. A significant amount of work has been conducted at the Air Force Research Lab (AFRL) to determine which part of the Xpatch program is the most time consuming. The finding is, the majority of the time (between 50% and 80%) is spent in executing the ray-tracing algorithms.

Among the commonly used shape primitives for ray tracing, Bézier patches stand out for being able to provide high modeling accuracy. On the other hand, computational expenses for ray/Bézier-patch intersection algorithms are especially high due to the nature and complexity of the algorithms. To tackle the dilemma, many attempts have been made in the past to develop efficient ray/Bézier-patch intersection algorithms. Recently proposed algorithms include Bézier clipping [NSK90], Chebyshev Boxing [FB94] and Bounding Volume Hierarchy [CSS97]. The Bézier clipping algorithm [NSK90] introduces an iterative geometric algorithm that finds all solutions of the ray-patch intersection problem up to an user definable accuracy, but it is hard to determine the iterative steps needed for a given patch. The Chebyshev Boxing algorithm [FB94] replaces an original patch by many bilinear approximating patches, which will be subdivided into pieces repeatedly until the original patch has been well approximated. The preprocessing procedure required by this algorithm uses a recursive method to subdivide a given patch. The bounding volume Hierarchy [CSS97] combines the previous two by first carrying out the preprocessing procedure in the Chebyshev Boxing algorithm and then calculating the tight bounding volumes. All these algorithms are based on recursive methods and therefore not suitable for hardware acceleration.

To speed up the ray tracing computation, significant efforts have also been made on the hardware side. The following works were reported in a survey conducted by Arvo and Kirk in 1989 [AK89]: LINKS-1, a 64-node (Intel

8086/8087) multicomputer that can be configured as a set of parallel pipelines to render a sequence of images; Kobayashi's work by distributing the world database among a set of intersection processors; and Goldsmith's work on a general-purpose multicomputers system for ray-tracing on a hypercube. More recent works include Kadi Bouatouch's exploitation [BB94] on using cache and virtual memory techniques that yielded a speedup of about 60 on a 64-node MIMD by reducing the complex message passing; Hyun-Joon Kin and Chong-Min Kyung's ring-structured multi-processors [KK96] which would allow a linear speedup (proportional to the number of processors used) by spreading the objects among the processors. Al the aforementioned hardware solutions are based on general-purpose architectures that are suitable for executing any kind of ray-object-intersection problems.

From the author's earlier research conducted at AFRL in Summer 1998, a ray/triangular-facet intersection algorithm optimized for hardware realization was developed and the corresponding hardware architecture was also proposed [Hung98]. As an extension of the author's previous work, the focus of this study is on accelerating the ray/Bézier-patch intersection computation in Xpatch. Again, efforts have been made on two aspects: 1) developing an algorithm that is suitable for hardware/software acceleration; 2) define the algorithm-specific system architecture for physical implementation in the future. As the results of this study, a new algorithm has been developed which is based on a set of sub-tasks including patch subdivision, hull classification, patch center calculation, ray/patch distance calculation, and sorting to find the four intersection points that are nearest to the ray origin. Based on the algorithm, a pipelined system architecture has been proposed. The number of coarse-grained pipeline stages was verified by extensive simulation. For performance estimation, the pipeline was refined, load-balanced and mapped into multiple TMS320C6701 digital signal processors (DSPs) and custom designed hardware for the sorting task. The results show that by reducing the grain scale of the pipeline and supporting the inherent parallelism of the computing tasks within the pipeline stages, an overall system throughput of over 1 million ray/patch intersections per second is achievable.

The rest of the report is organized as follows: Section 2 introduces the background and the computation involved with the proposed Ray/Bézier-Patch intersection algorithm, as well as the verification of the algorithm; Section 3 describes the functional blocks required for executing the algorithm, and proposes the overall system architecture without explicit specifications on software/hardware boundaries; Section 4 describes the performance estimation based on implementation models relying on off-the-shelf DSPs and limited ASIC for an identified computational bottleneck; Section 5 concludes the report.

2. Algorithm and Computation for Ray/Bézier-Patch Intersection

2.1 Background of the Proposed Algorithm

The scene of a ray intersects with a Bézier patch in a 3D space is illustrated by Fig. 2-1. Theoretically, there are 18 possible intersection points when a ray piercing a bicubic patch [KAJ82]. From a practical standpoint, our interest here is to find the four intersection points that are nearest to the ray origin. Also, our study has been focused on bicubic Bézier patch, which is frequently referred to as 'patch' or 'Bézier patch' in this report for the sake of simplicity.

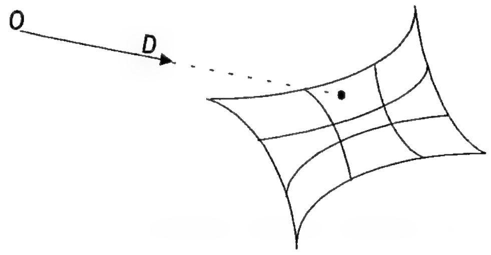


Figure 2-1 Ray/Bézier-Patch intersection

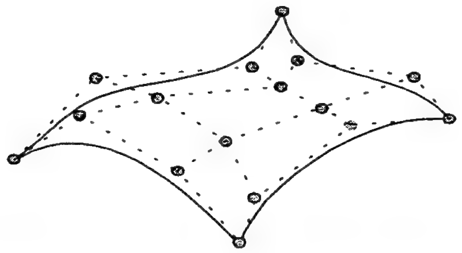


Figure 2-2 Convex hull of bicubic Bézier patch

One of the major difficulties associated with the ray/patch intersection problem is that there are no exact solutions to the problem. As the consequence, we have to look for approximated solutions. A parametric bicubic Bézier patch is characterized by 16 control points that are connected as a mesh as shown in Fig. 2-2. The *property of convex hull* states that the Bézier patch must lie inside its convex hull - the polyhedron formed by connecting the patch's control points [WW98]. With this property, one can conclude that, as the necessary (but not sufficient) condition for a ray to intersect with a given patch, the ray must hit the patch's convex hull first. Note that, the ray may still miss the patch even it hits the convex hull, as the ray may pass through the clearance between the hull boundary and the patch surface. The probability of this phenomenon, however, will become lower and lower as the size of the patch gets smaller and smaller. When a ray hits the convex hull of a sufficiently small patch, the center of the patch can be considered as a good approximation of the ray/patch intersection point.

To determine whether or not the ray hits the convex hull of a given patch, we took an approach based on the concept of *virtual projection* which is illustrated by Fig. 2-3 through 2-6. Imagine that a ray is determined by the intersection line of two planes U and V who are perpendicular to each other, and that a projection plane perpendicular to the ray direction is placed in front of the ray and a testing patch. We then project the two planes U and V together with the ray and the testing patch onto the projection plane. What we will see on that plane is 2D coordinate system with 16 points scattered inside. The two axes of the coordinate system are projected from the two planes U and V; the origin of the coordinate system is projected from of the ray, and the 16 points are projected from the control points of the patch. The figures also show that, in the projected 2D coordinator system, if the 16 projected control points are located on the same side of either one of the two axes, the ray must have missed the convex hull of the testing patch (refer to Fig. 2-4 and Fig. 2-5); otherwise the ray hits the convex hull (refer to Fig. 2-6).

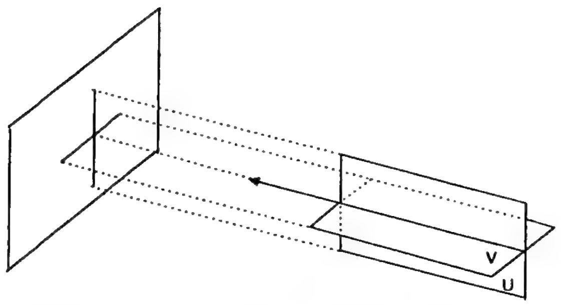


Figure 2-3. The concept of virtual projection

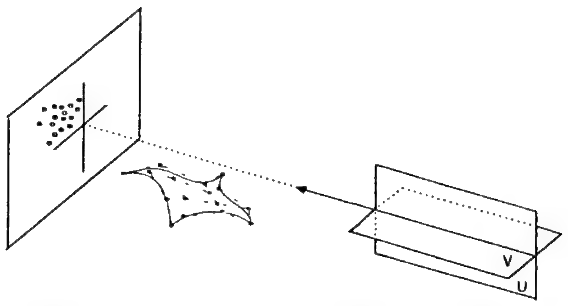


Figure 2-4. Case 1: The ray misses the patch

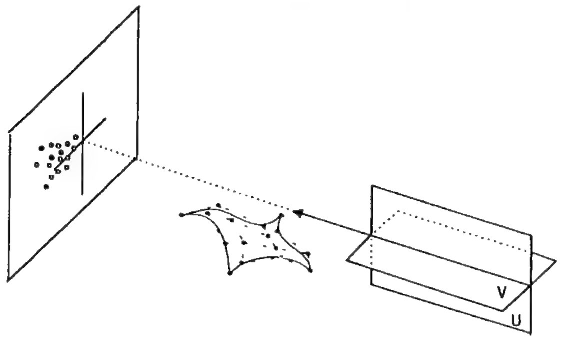


Figure 2-5. Case 2: The ray misses the patch

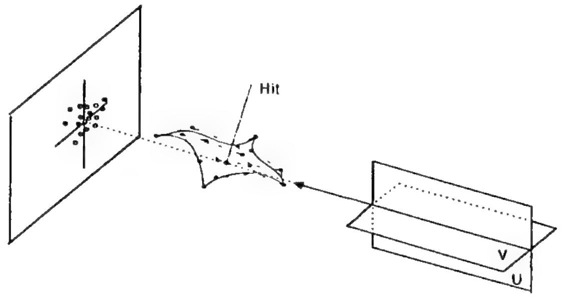


Figure 2-6. Case 3: The ray hits the convex hull of the patch

Based on the above discussion, we outline the idea behind the ray/Bézier-patch intersection algorithm developed in this study:

- Divide an input patch into 16 sub-patches, each is characterized by 16 control points,
- Find all sub-patches that are possibly intersected by the ray, based on the property of convex hull and the concept of virtual projection,
- From all sub-patches obtained from the last step, find four sub-patches that are most close to the origin of the ray,
- Subdivide the four sub-patches again to obtain 16 smaller patches, and repeat the previous steps until the approximated solutions (the centers of four tiny sub-patches) satisfy the desired resolution.

Mathematical notations and detailed computations involved in our algorithm are discussed in the following subsections.

2.2 Mathematical Notations

Mathematical notations used in this report, for point, ray, plane and Bézier patch in a 3D space, are listed below:

A point:

$$\mathbf{P} = (x,\ y,\ z)$$

A ray:

$$\mathbf{R} = \mathbf{O} + \mathbf{D} * t \tag{1}$$

where the origin (**O**) and direction (**D**) of the ray are defined by $(O_x,\ O_y,\ O_z)$ and $(D_x,\ D_y,\ D_z)$ respectively.

A plane:

$$A * x + B * y + C * z + D = 0 \tag{2}$$

where (A, B, C) is the normal of the plane.

A general nonrational Bézier patch:

$$\mathbf{Q}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{P}_{i,j} B_{j,m}(u) B_{i,n}(v) w_{i,j} \quad \text{for } 0 \le u, v \le 1 \tag{3}$$

where $\mathbf{P}_{i,j} = (x_{i,j},\ y_{i,j},\ z_{i,j})$ are the Bézier patch's control points with associate weights $w_{i,j} = 1$ in this study, and the blending functions are the Bernstein polynomials as shown below:

$$B_{j,m}(u) = \binom{m}{j} u^j (1-u)^{m-j}, \quad B_{i,n}(v) = \binom{n}{i} v^i (1-v)^{n-i}, \quad \binom{n}{j} = \frac{n!}{j!(n-j)!}, \quad \binom{n}{i} = \frac{n!}{i!(n-i)!}$$

A Bézier patch can be also presented in its matrix format. That is,

$$\mathbf{Q}(u, v) = [\mathbf{U}][\mathbf{M}][\mathbf{P}][\mathbf{M}]^t\ \mathbf{V} \tag{4}$$

where,

$[\mathbf{U}] = \begin{bmatrix} u^m & u^{m-1} & \cdots & 1 \end{bmatrix}$, $[\mathbf{V}] = \begin{bmatrix} v^n & v^{n-1} & \cdots & 1 \end{bmatrix}^t$, $[\mathbf{P}]$ is the control point matrix. $[\mathbf{M}]$ is the Bézier basis matrix, and $[\mathbf{M}]^t$ is the transpose of $[\mathbf{M}]$.

The bicubic Bézier patch is a special case of (4) with $n = m = 3$. That is,

$$\mathbf{Q}(u, v) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} [\mathbf{M}][\mathbf{P}][\mathbf{M}]^t \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix} \tag{5}$$

where

$$[M] = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix} \tag{6}$$

2.3 Computation for Generating the Two Planes that Determine a Given Ray

From the discussion in Section 2.1, given a ray $\mathbf{R} = \mathbf{O} + \mathbf{D} * t$, our ray/Bézier-patch intersection algorithm needs to find the two planes U and V that determine the ray. That is, to find

$$A_U * x + B_U * y + C_U * z + D_U = 0 \tag{7.a}$$

and $$A_v * x + B_v * y + C_v * z + D_v = 0 \tag{7.b}$$

from the parameters of the given ray. Let the two planes U and V to be perpendicular to each other as illustrated in Fig. 2-7 (where **N*u*** and **N*v*** are normals of the planes U and V respectively, and **D** is the direction of the ray), we have:

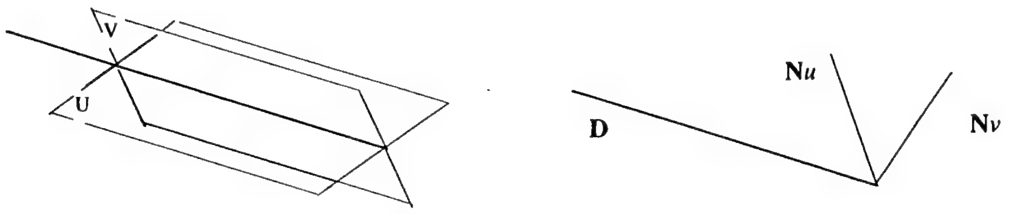


Figure 2-7 Plane Generation

$$(A_u \ B_u \ C_u) \equiv (O_x \ O_y \ O_z) \times (D_x \ D_y \ D_z) \tag{8.a}$$

$$(A_v \ B_v \ C_v) \equiv (A_u \ B_u \ C_u) \times (D_x \ D_y \ D_z) \tag{8.b}$$

$$D_u \equiv -(A_u \ B_u \ C_u) \bullet (O_x \ O_y \ O_z) \tag{8.c}$$

$$D_v \equiv -(A_v \ B_v \ C_v) \bullet (O_x \ O_y \ O_z) \tag{8.d}$$

The above computation can be simplified. When one of the components of the ray direction vector **D** is zero (in this case we let D_x to be zero), parameters of the planes U and V can be determined by:

$$\mathbf{N}u = (1 \quad 0 \quad 0) \times \mathbf{D} = \ 0\hat{x} - D_z\hat{y} + D_y\hat{z} = (0 \quad -D_z \quad D_y) \tag{9.a}$$

$$\mathbf{N}v = (1 \ 0 \ 0) \tag{9.b}$$

$$Du = -\mathbf{N}u \bullet \mathbf{O} = D_z \cdot O_y - D_y \cdot O_z \tag{9.c}$$

$$Dv = -\mathbf{N}v \bullet \mathbf{O} = -O_x \tag{9.d}$$

If none of the components of **D** is zero, then parameters of the two planes can be determined by:

$$\mathbf{N}u = \begin{pmatrix}0 & 0 & 1\end{pmatrix} \times \mathbf{D} = \begin{pmatrix}-D_y & D_x & 0\end{pmatrix} \tag{10.a}$$

$$\mathbf{N}v = \mathbf{N}u \times \mathbf{D} = \begin{pmatrix}D_x D_z & D_y D_z & -D_y^2 - D_z^2\end{pmatrix} \tag{10.b}$$

$$Du = -\mathbf{N}u \bullet \mathbf{O} = D_y O_x - D_x O_y \tag{10.c}$$

$$Dv = -\mathbf{N}v \bullet \mathbf{O} = -(Nv._x \cdot O_x + Nv._y \cdot O_y + Nv._z \cdot O_z) \tag{10.d}$$

Note that in the above formula, **N*u*** and **N*v*** are orthogonal to the ray direction, and at the same time **N*u*** and **N*v*** are orthogonal to each other. The overall computation required for generating the two planes U and V are summarized in the algorithm below.

Plane Generation Algorithm:

Step 1: If $D_x \neq 0$ goto step2, else

$\mathbf{N}u = (0 \quad -D_z \quad D_y)$; $\quad \mathbf{N}v = (1 \quad 0 \quad 0)$; $Du = D_z \cdot O_y - D_y \cdot O_z$; $Dv = -O_x$; return

Step 2: if $D_y \neq 0$ goto step 3, else

$\mathbf{N}u = (D_z \quad 0 \quad -D_x)$; $\quad \mathbf{N}v = (0 \quad 1 \quad 0)$; $Du = D_x \cdot O_z - D_z \cdot O_x$; $Dv = -O_y$; return

Step 3 : if $D_z \neq 0$ goto step 4, else

$\mathbf{N}u = (-D_y \quad D_x \quad 0)$; $\quad \mathbf{N}u = (0 \quad 0 \quad 1)$; $Du = D_y O_x - D_x O_y$; $Dv = -O_z$; return

Step 4: $\mathbf{N}u = (0 \quad 0 \quad 1) \times \mathbf{D} = \left(-D_y \quad D_x \quad 0\right)$; $\mathbf{N}v = \mathbf{N}u \times \mathbf{D} = \left(D_x D_z \quad D_y D_z \quad -D_y^2 - D_z^2\right)$;

$Du = -\mathbf{N}u \bullet \mathbf{O} = D_y O_x - D_x O_y$; $Dv = -\mathbf{N}v \bullet \mathbf{O} = -(Nv._x \cdot O_x + Nv._y \cdot O_y + Nv._z \cdot O_z)$; return

2.4 Computation for Patch Subdivision

It was discussed in Section 2.1 that, in order to obtain approximated solutions for the ray/patch intersection problem, we have to divide an input patch in to smaller sub-patches. The patch subdivision computation described here is based on the de Casteljau algorithm [FOL97] which has the advantage that, while a patch is being divided, control points of all the sub-patches are generated directly. The computation takes two steps to divide a given patch into four sub-patches. In the first step, we fix the variable v in Equation (5) and apply the left and right Bézier division matrices D_B^L and D_B^R to the given Bézier patch [**P**] to obtain the left and right Bézier sub-patches P_B^L and P_B^R respectively. That is,

$$P_B^L = D_B^L [\mathbf{P}] \tag{11}$$

and

$$P_B^R = D_B^R [\mathbf{P}] \tag{12}$$

In the second step, we fix the variable u in Equation (5) and apply the same method to the sub-patches P_B^L and P_B^R obtained in the previous step, to obtain the four sub-patches P_B^{LL}, P_B^{LR}, P_B^{RL}, and P_B^{RR} (also denoted as P00, P01,P10, and P11) respectively. That is,

$$P_B^{LL} = D_B^L \; P_B^L, \; P_B^{LR} = D_B^R \; P_B^L, \; P_B^{RL} = D_B^L \; P_B^R \text{ and } P_B^{RR} = D_B^R \; P_B^R. \tag{13}$$

The given Bézier patch [**P**], the left and right Bézier division matrices D_B^L and D_B^R are defined by

$$[\mathbf{P}] = \begin{bmatrix} p00 & p01 & p02 & p03 \\ p10 & p11 & p12 & p13 \\ p20 & p21 & p22 & p23 \\ p30 & p31 & p32 & p33 \end{bmatrix}, \; D_B^L = \frac{1}{8}\begin{bmatrix} 8 & 0 & 0 & 0 \\ 4 & 4 & 0 & 0 \\ 2 & 4 & 2 & 0 \\ 1 & 3 & 3 & 1 \end{bmatrix}, \text{ and } D_B^R = \frac{1}{8}\begin{bmatrix} 1 & 3 & 3 & 1 \\ 0 & 2 & 4 & 2 \\ 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 8 \end{bmatrix} \tag{14}$$

respectively.

2.5 Computation for Determining If a Given Ray Hits the Convex Hull of a Given Patch – the *Hull Classification*

In Section 2.1 we introduced the idea of virtual projection, and showed how to determine whether a given ray hits the convex hull of a given patch by looking at the allocation of the projected patch control points in the projected 2D coordinate system. Note that, the problem of determining the position of a projected control point in the projected 2D coordinate system (specifically, on which side of the axes the point is located) is equivalent to the problem of determining the relative position of an original control point with respect to the original planes (U and V) in the 3D space (specifically, on which side of the planes the point is located). The actual computation thus can be carried out in the 3D space as this will be more efficient.

Recall that in a 3D space, the distance of a point **P** at (*x, y, z*) to a given plane $A*x+B*y+C*z+D=0$ is given by

$$d=\frac{Ax+By+Cz+D}{\sqrt{A^2+B^2+C^2}} \tag{15}$$

This distance is either negative or positive depending on at which side of the plane the point **P** is located (the distance is zero if **P** is on the plane). Since we are only interested in the sign of *d*, which is solely determined by the numerator of the right-hand side of the Equation (15), we can define the *sign distance* as shown below:

$$d'=Ax+By+Cz+D\,, \tag{16}$$

and in matrix form,

$$d'=\begin{bmatrix}A & B & C & D\end{bmatrix}\begin{bmatrix}\mathbf{P}'\\1\end{bmatrix}=\begin{bmatrix}A & B & C & D\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix} \tag{17}$$

Once the planes U and V are obtained, we can determine the patch's position by evaluating the sign distances between the control points and the planes. To be specific, for either plane U or V, if signs of the sixteen sign distances are either all positive or all negative, the ray misses the patch. If this is untrue for both planes, the ray hits the convex hull of the patch and hence it may also intersect with the patch itself. Note that evaluation processes against the planes U and V can be done in parallel.

The explicit computation for calculating the 16 sign distances with respect to the planes U and V are:

$$\begin{bmatrix}du0 & du1 & \cdots & du14 & du15\end{bmatrix}=\begin{bmatrix}Au & Bu & Cu & Du\end{bmatrix}\begin{bmatrix}p_{00_x} & p_{01_x} & \cdots & p_{32_x} & p_{33_x}\\ p_{00_y} & p_{01_y} & \cdots & p_{32_y} & p_{33_y}\\ p_{00_z} & p_{01_z} & \cdots & p_{32_z} & p_{33_z}\\ 1 & 1 & 1 & 1 & 1\end{bmatrix} \tag{18}$$

and

$$\begin{bmatrix}dv0 & dv1 & \cdots & dv14 & dv15\end{bmatrix}=\begin{bmatrix}Av & Bv & Cv & Dv\end{bmatrix}\begin{bmatrix}p_{00_x} & p_{01_x} & \cdots & p_{32_x} & p_{33_x}\\ p_{00_y} & p_{01_y} & \cdots & p_{32_y} & p_{33_y}\\ p_{00_z} & p_{01_z} & \cdots & p_{32_z} & p_{33_z}\\ 1 & 1 & 1 & 1 & 1\end{bmatrix} \tag{19}$$

respectively.

Two vectors $Su=\begin{bmatrix}Su0 & Su2 & \cdots & Su14 & Su15\end{bmatrix}$ and $Sv=\begin{bmatrix}Sv0 & Sv2 & \cdots & Sv14 & Sv15\end{bmatrix}$are defined to hold the signs of the sign distances obtained from (18) and (19), respectively. As an example,

$$Su0 = sign(du0) = \begin{cases} -1 & if\ du0 < 0 \\ 0 & if\ du0 = 0 \\ 1 & if\ du0 > 0 \end{cases} \qquad (20)$$

For both *Su* and *Sv*, if the sum of all elements is not equal to -16 or 16, the ray may intersect with the patch and thus the patch needs to be further divided into smaller sub-patches for more accurate approximation.

2.6 Computation for Determining the Center of the Patch

Based on the discussion given in Section 2.1, it is necessary to find out the center of a given patch for two reasons: 1) a ray/patch intersection point is approximated by the center of a sufficiently small patch; 2) in the process of finding the approximated intersection points, our algorithm must sort out four sub-patches (from all sub-patches possibly hit by the ray) that are most close to the origin of the ray, where the "closeness" is measured by the distance between the origin of the ray and the center of a given patch.

For a given patch **Q** (*u,v*) defined in Equation (4), its center can be obtained by the following computation:

$$\mathbf{Q}(1/2,\ 1/2) = [\mathbf{U}][\mathbf{M}][\mathbf{P}][\mathbf{M}]^t\ \mathbf{V} \qquad (21)$$

where

$$[\mathbf{U}] = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix},\ [\mathbf{V}] = \begin{bmatrix} v^3 & v^2 & v & 1 \end{bmatrix}^t,\ u = v = 1/2$$

for a bicubic Bézier patch.

For each component of **Q**, we have:

$$Q(1/2,1/2)_x = [U][M][P]_x[M]^t[V] \qquad (22.a)$$

$$Q(1/2,1/2)_y = [U][M][P]_y[M]^t[V] \qquad (22.b)$$

$$Q(1/2,1/2)_z = [U][M][P]_z[M]^t[V] \qquad (22.c)$$

Based on the above, we can derive the formula below for efficient computation:

$$\begin{aligned} \mathbf{Q}(\mathrm{u},\mathrm{v}) = {} & (1/64)*(\mathbf{p}_{00} + \mathbf{p}_{03} + \mathbf{p}_{30} + \mathbf{p}_{33}) \\ & +(3/64)*(\mathbf{p}_{01} + \mathbf{p}_{02} + \mathbf{p}_{10} + \mathbf{p}_{13} + \mathbf{p}_{20} + \mathbf{p}_{23} + \mathbf{p}_{31} + \mathbf{p}_{32}) \\ & +(9/64)*(\mathbf{p}_{11} + \mathbf{p}_{12} + \mathbf{p}_{21} + \mathbf{p}_{22}) \end{aligned} \qquad (23)$$

2.7 Computation for Determining the Distance Between the Origin of the Ray and the Center of a Given Patch

Given two points $\mathbf{P}_1(x_1, y_1, z_1)$ and $\mathbf{P}_2(x_2, y_2, z_2)$ in a 3D space, the distance between them is defined by:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \qquad (24)$$

To simplify the computation, we redefine the distance as:

$$d = abs(x_1 - x_2) + abs(y_1 - y_2) + abs(z_1 - z_2) \qquad (25)$$

Whether the ray runs into or runs away from the patch is determined by the angle between the ray direction **D** and the vector from the ray origin **O** to the patch center **C**, as illustrated by Fig. 2-8. Specifically, if $\cos\theta = \dfrac{OC \cdot D}{|OC||D|} < 0$ the ray runs away from the patch, otherwise the ray runs into (i.e., hits) the patch.

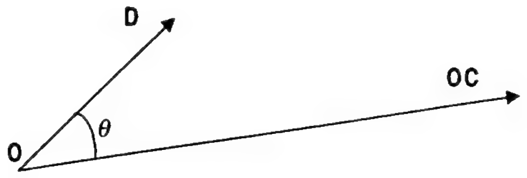


Figure 2-8. Ray direction **D** and the vector from the ray origin **O** to the patch center **C**

Again, since we are only interested in the sign of $\cos\theta$, for obtaining the direction information, all we need to calculate is **OC·D** and, with which we can define the *signed ray/patch distance* as shown below. Where d is defined in Equation (25).

$$d_p = \begin{cases} d & \text{if } \mathbf{OC}\cdot\mathbf{D} > 0 \\ -1 & \text{if } \mathbf{OC}\cdot\mathbf{D} < 0 \end{cases} \tag{26}$$

When **OC·D** < 0, d_p is force to be '-1' which indicates that the ray runs away from the patch.

2.8 Computation for Finding the Four Sub-Patches that Most Close to the Ray Origin – the *Culler*

In our patch/ray intersection algorithm, an input patch is first divided into 16 sub-patches; each sub-patch is then evaluated to see if its convex hull intersects with the ray (refer to Section 2.1). From all sub-patches that passed the evaluation (i.e., their convex hull were hit by the ray), four with the shortest ray/patch distance (defined in the previous section) will be sort out for further approximation. The computation required by the sorting task is to execute the *quick sort* algorithm, which was chosen because given an input array of size *n*, the *quick sort* runs faster than other sorting algorithms with $\Theta(n \lg n)$ run time. Fig. 2-9 compares the efficiency of the quick sort algorithm with another two popular sorting algorithms – the *heap sort* and the *merge sort*. Clearly, the *quick sort* is the best choice.

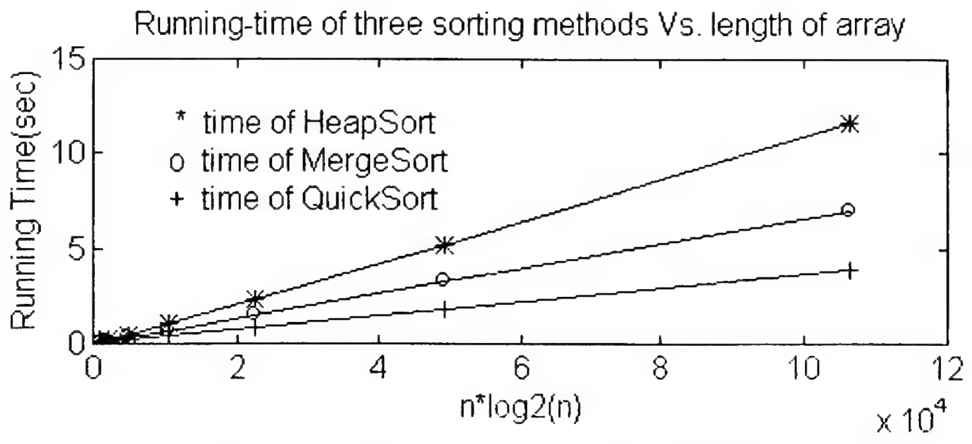


Figure 2-9 Comparison of different sorting algorithms

2.9 Summary on the Ray/ Bézier-patch Intersection Algorithm

Based on the previous discussion, we now summarize the ray/Bézier-patch intersection algorithm as shown below, and its corresponding flow chart is given in Fig. 2-10.

Algorithm for Finding Ray/ Bézier-patch Intersection:

step1: Subdivide the input patch into four sub-patches (see Section 2.4, Formula 11-14) and put them into level-1 list which can store at most four sub-patches; generate planes U and V from the ray parameters (see Section 2.3, Plane Generation Algorithm).

step2: If the desired number of iteration is not yet reached, subdivide the each patch in the level-1 list into four sub-patches (see Section 2.4, Formula 11-14) and store them into level-2 list which can store at most 16 sub-patches, otherwise go to Step 5.

step3: Evaluate each patch in level-2 list by *hull classification* (see Section 2.5, Formula 18-20); calculate the centers of the patches (see Section 2.6, Formula 23) and their distances to the origin of the ray (see Section 2.7, Formula 26).

step4: Sort the patches in level-2 list to find the four patches closest to the ray origin (see Section 2.8) and put them into the level-1 list; back to step2.

step5: Output the centers of the four sub-patches (if they are close to each other enough, then combine them into single point).

With this algorithm, assuming n iterations are repeated and there is an intersection point, the patch containing this point will be subdivided by n times, each time into four small ones. Therefore the sub-patches obtained at the end of the iteration will be $1/4^n$ of the original patch size. As long as the input patch is reasonably small, the four final sub-patches will converge to a tiny area and the center of which will be considered as the intersection point.

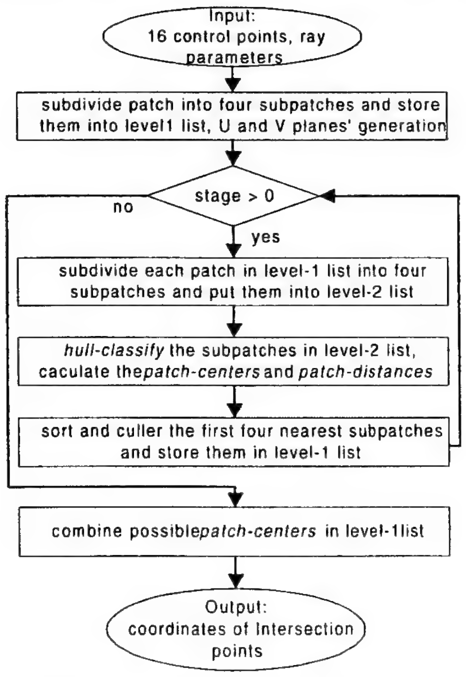

Input:
16 control points, ray parameters
subdivide patch into four subpatches and store them into level1 list, U and V planes' generation
stage > 0
no
yes
subdivide each patch in level-1 list into four subpatches and put them into level-2 list
hull-classify the subpatches in level-2 list, caculate the*patch-centers* and *patch-distances*
sort and culler the first four nearest subpatches and store them in level-1 list
combine possible*patch-centers* in level-1list
Output:
coordinates of intersection points

Figure 2-10 Flow chart of the algorithm

2.10 Verification of the Proposed Algorithm

In order to verify the proposed ray/Bézier-patch intersection algorithm we developed a user-friendly interface. With which arbitrary rays and Bézier patches can be specified by the user, and the algorithm is then tested extensively in software. Using the teapot patches as the benchmark, we also compared the results obtained from our algorithm with those from Nishita's algorithm [NSK90] (which is based on the Bézier clipping technique). We found that the intersection points generated by the two algorithms are close enough with respect to the accuracy defined by the Bézier clipping approach. The following case is picked arbitrarily as an example, where the input parameters for patch and ray are

Patch: **P** =

{ <-0.6,0.0,-0.6>, <-0.6,0.8,-0.2>, <-0.6,0.8,0.2>, <-0.6,0.0,0.6>,
<-0.2,0.2,-0.6>, <-0.2,0.4,-0.2>, <-0.2,0.4,0.2>, <-0.2,0.2,0.6>,
<0.2,0.2,-0.6>, <0.2,0.4,-0.2>, <0.2,0.4,0.2>, < 0.2,0.2,0.6>,
<0.6,0.0,-0.6>, <0.6,0.8,-0.2>, <0.6,0.8,0.2>, <0.6,0.0,0.6>}

ray: **R** = {<0.7,0.55,0.05>, <-1.0,0.0,0.0>}

respectively. With the input parameters shown above, both algorithms find two intersection points after ten iteration stages. The difference between the solutions from the two algorithms is 10^{-4} which is the user-defined accuracy in Nishita's algorithm. Here the difference is defined by $\mathrm{Max}(|x-x'|,|y-y'|,|z-z'|)$, where (x, y, z) and (x', y', z') are two correspondent points in the two algorithms. The detailed results for the given case are shown in Table 1.

The proposed algorithm (x,y,z)	Bézier clipping algorithm (x',y',z') EPISLON=10^{-4}	Difference Max(|x-x'|, |y-y'|, |z-z'|)
0.520605468750 0.550060024155 0.050097656250	0.520467170854 0.549999999993 0.050000000000	10^{-4}
-0.520605468750 0.550060024155 0.050097656250	0.520467170854 0.549999999993 0.050000000000	10^{-4}

Table 2-1 Comparison of the results from the proposed algorithm and Nishita's algorithm (For both algorithms, number of iterations: 10; number of intersection points obtained: 2)

3. Functional Building Blocks and the System Architecture

In this section, we define the functional building blocks required by the ray/Bézier-patch intersection algorithm discussed in Section 2. Based on which, a pipelined system architecture is proposed. The purpose here is to specify the functionality of each building block as well as the structure and operation of the overall system, rather than the hardware/software boundaries and implementation details.

In order to compare the relative computing load of the subtasks required by the proposed ray/Bézier-patch intersection algorithm, a set of C function calls are developed and listed in Table 3-1.

Names of the Function Calls	Tasks of the Function Calls
PlaneGenerator	to generate the two planes U and V from a given ray
PatchSubdividor	to divide an input patch into four sub-patches
PatchXPlane	to execute the task of *hull classification* in Step 3 of the algorithm
GetPatchCenter	to calculate the center of a given patch
GetDistance	to calculate the distance from the patch center to the ray origin
QuickSort	to sort out 4 sub-patches nearest to the ray origin from 16 candidates

Table 3-1. Function calls for the ray/Bézier-patch intersection algorithm

Execution times for the function calls are obtained and plotted in Fig. 3-1. These data reveal the load distribution among the different computing tasks and show the computational bottlenecks. They serve as guidelines for load balancing (at the stage of defining the functional building blocks and system architecture) and hardware/software partitioning (at the stage of physical implementation, e.g., the sorting task may need ASIC acceleration).

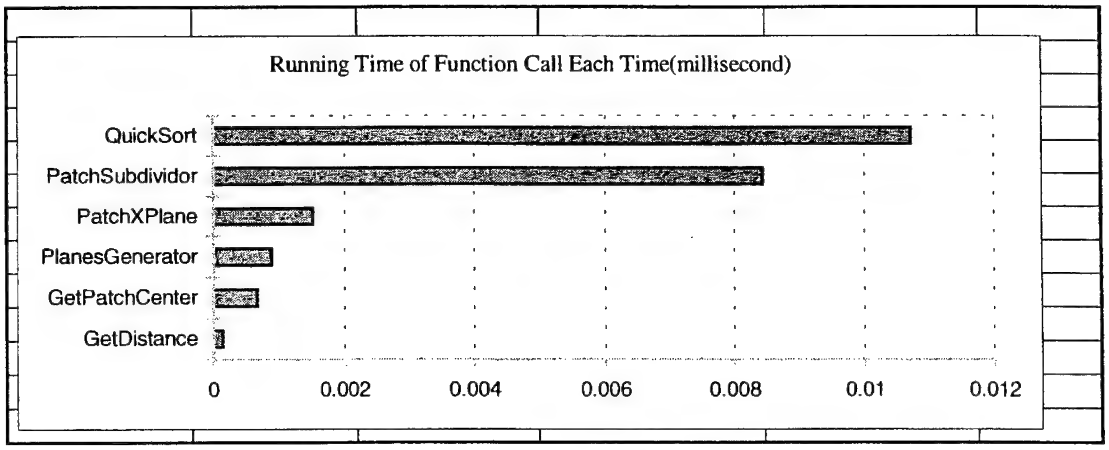


Figure 3-1. Execution time of for each function call under the following condition:

- Platform: PC with 350MHz Intel Pentium II-MMX processor
- Operating system: Windows NT 4.0 with networking activities disabled
- Compiler: Visual C++ 6.0 with the *Maximize Speed* option on

3.1 M0: the Plane Generation Module

This functional block corresponds to the function call *PlaneGenerator.* Its task is to generate the two planes U and V from the parameters (origin and direction) of a given ray, which is required by Step 1 of the proposed algorithm. Block diagram of the module is given in Fig. 3-2.

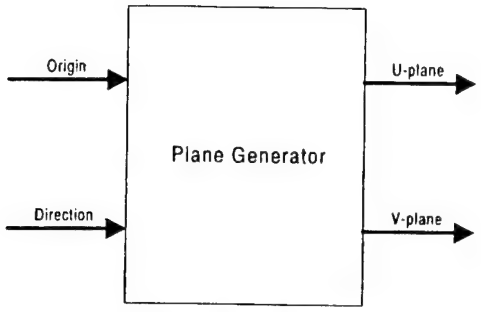


Figure 3-2. M0: the Plane Generation Module

3.2 M1: the Patch Subdivision Module

This functional block corresponds to the function call *PatchSubdividor*. It divides an input patch into four sub-patches, and is used in Step 1 and Step 2 of the proposed algorithm. The block diagram in Fig. 3-3 shows that, the module consists of three identical submodules named *split-in-half*, each of them splits its input patch into two pieces by executing two matrix multiplications (refer to Formula 11 – 13 in Section 2.4).

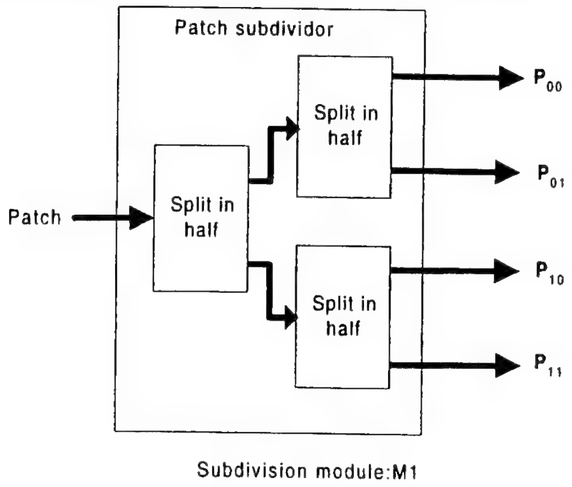
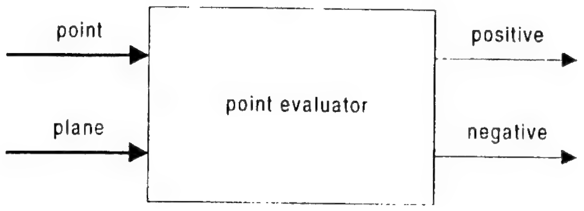


Figure 3-3. M1: the Patch Subdividsion Module

3.3 Submodule: the Point Evaluator

This submodule calculates the *sign-distance* (refer to Formula 16 in Section 2.5) from a single point (a control point of a patch) to one of the two planes (U or V) that determine a ray, and outputs the obtained sign information which indicates the point's position with respect to the given plane. Block diagram and functionality of the submodule is given in Fig. 3-4.



positive	negative	meaning
0	0	point is on the plane
0	1	the distance is negative
1	0	the distance is positive
1	1	prohibited

Figure 3-4. Submodule: the Point Evaluator

3.4 Submodule: the Patch Evaluator

This submodule corresponds to the function call *PatchXPlane.* Its block diagram is given in Figure 3-5. The input labeled 'patch' contains the sixteen control points of a given patch. Using 16 *point evaluators* (defined in the previous subsection) in parallel, the *patch evaluator* is able to determine the input patch's position with respect to a given plane (U or V). The submodule's output signal *miss* becomes active when all the 16 control points are on the same side of the plane, which indicates that the ray missed the patch; as discussed in Section 2.5.

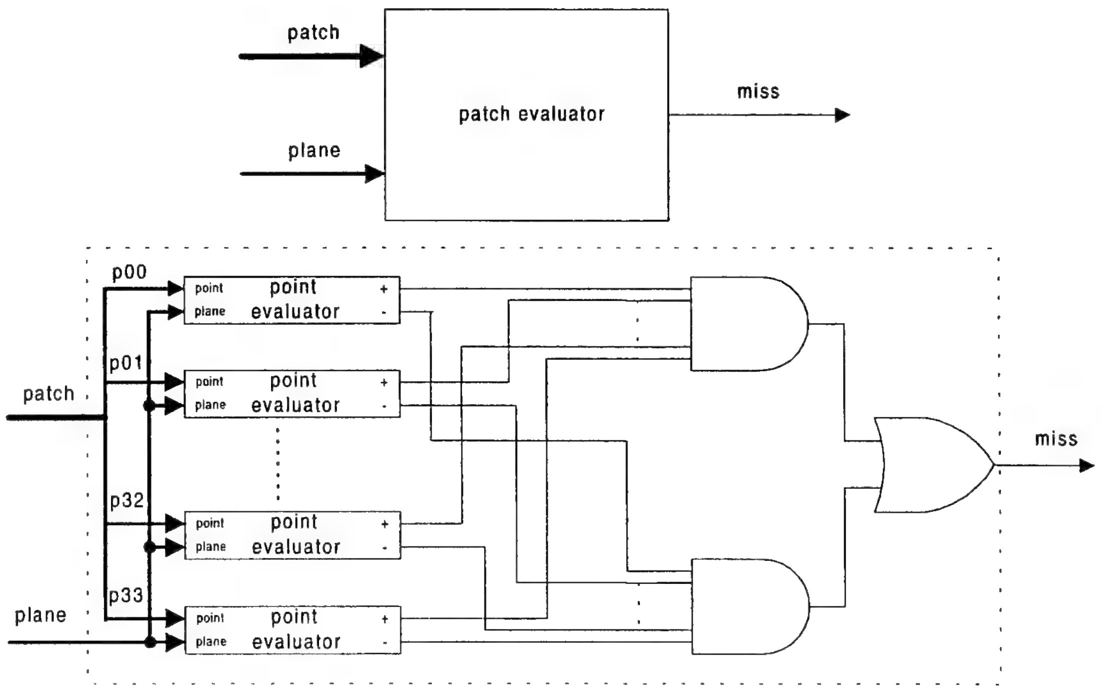
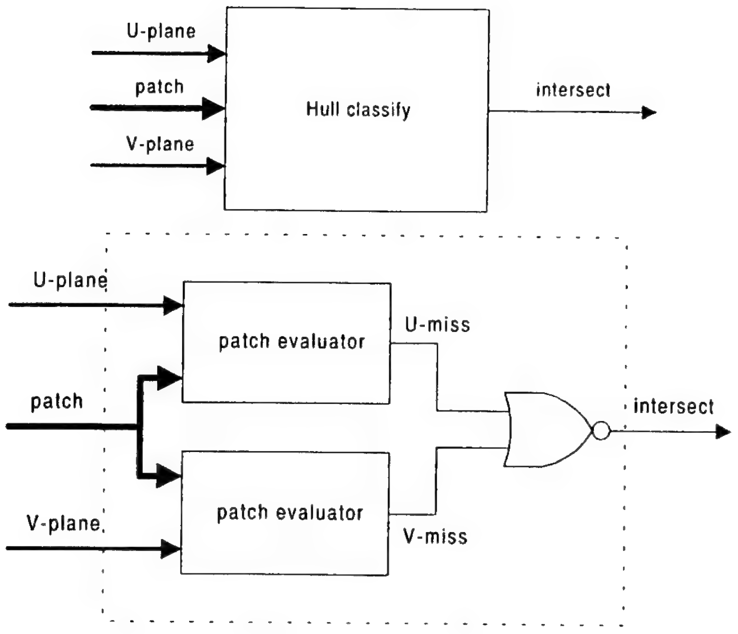


Figure 3-5. Submodule: the Patch Evaluator

3.5 Submodule: the Hull Classifier

Built upon two *patch evaluators* discussed in the previous subsection, this submodule concurrently evaluates the position of the input patch against the two planes U and V, thus completes the task of hull classification (refer to Section 2.5) in Step 3 of the proposed algorithm. The schematic and functionality is given in Figure 3-3. Block diagram and functionality of the submodule is given in Fig. 3-6.



U-miss	V-miss	intersect
0	0	1
0	1	0
1	0	0
1	1	0

Figure 3-6. Submodule: the Hull Classifier

3.6 Submodule: the Patch Center

As part of the computing task in Step 3 of the proposed algorithm, this submodule calculates the center of a given patch. It corresponds to the function call *GetPatchCenter*. The computation involved is described in Section 2.6. The block diagram of the submodule is given in Fig. 3-7.

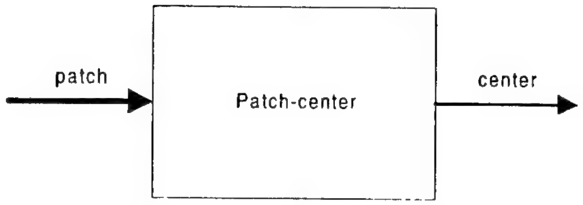


Figure 3-7. Submodule: the Patch Center

3.7 Submodule: the Ray/Patch Distance

Corresponds to the function call *GetDistance*, this submodule calculates the distance from the patch center to the ray origin, to fulfill part of the computing task in Step 3 of the proposed algorithm. Its computation is described in Section 2.7, and block diagram is given in Fig. 3-8.

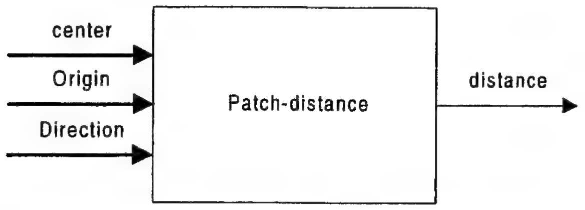


Figure 3-8. Submodule: the Ray/Patch Distance

3.8 M2: the Intersection Module

This functional building block combines the submodules *hull classifier*, *patch center* and *ray/patch distance* discussed in the previous subsections to accomplish all the computing tasks involved in Step 3 of the proposed algorithm. Block diagram of the module is shown in Fig. 3-9. When entering the module, the input patch will be immediately evaluated by the *hull classifier* to see if there is a possible ray/patch intersection. Once confirmed, the center of the input patch and the ray/patch distance will be calculated successively, and the input patch will be passed to the *culler* (refer to Section 2.8) along with the corresponding ray/patch distance (labeled as 'distance' in Fig. 3-9).

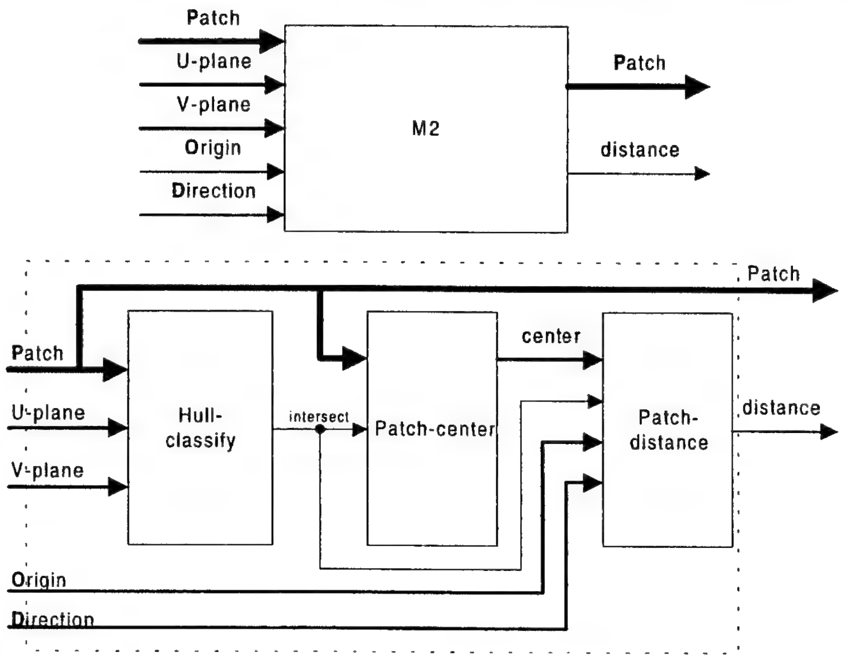


Figure 3-9. M2: the Intersection Module

3.9 The Culler Module

The task of this module is to execute Step 4 of the proposed Ray/Bézier-Patch intersection algorithm. It takes 16 sub-patches and their corresponding ray/patch distances as inputs, and outputs four sub-patches that are most close to the ray origin. To increase the sorting efficiency and reduce the memory requirement, the ray/patch distance is used as the sorting key. In order to identify its corresponding patch, each sorting key must carry a tag, as illustrated by Fig. 3-10. Where the ray/patch distance is in IEEE 754 single precision floating-point format (bit 0 to 31); a 4-bit tag *Patch Index* is padded to the beginning of the data (bit 32 to 35).

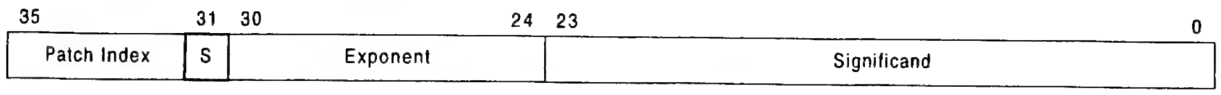
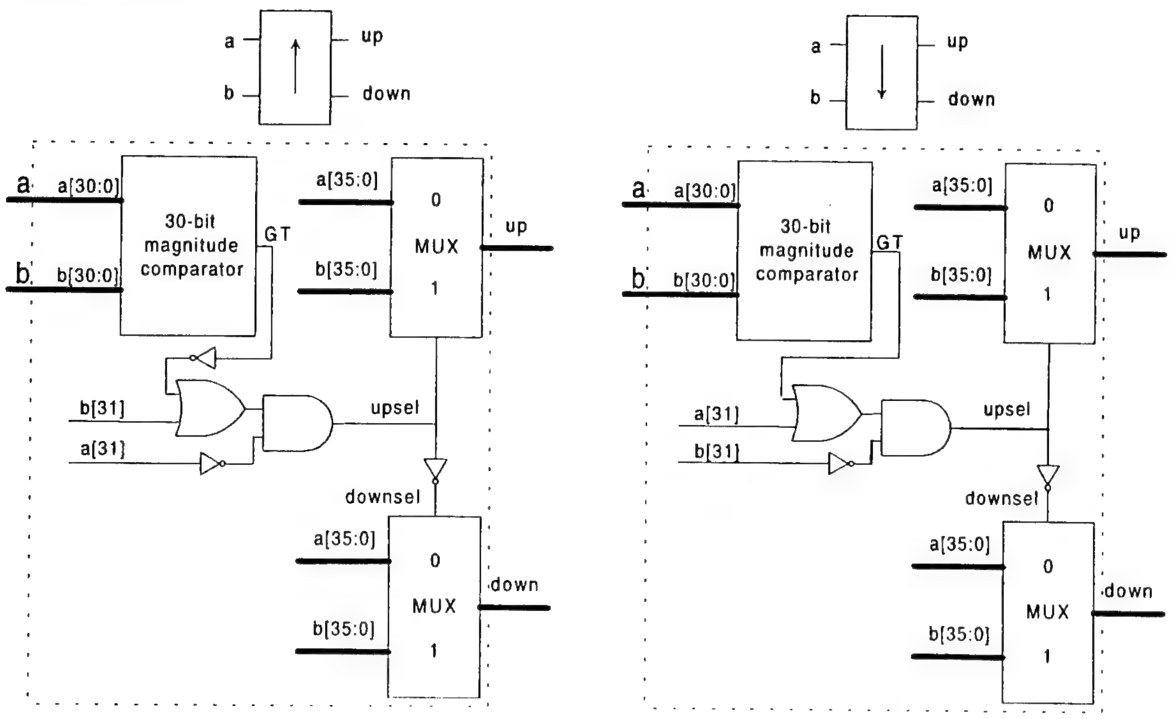


Figure 3-10. Format of the sorting key

Two basic functional units are defined for the sorting task: the *upsort* and the *downsort*. Both units take two sorting keys *a* and *b* as inputs, and have two outputs labeled *up* and *down*. For the unit *upsort*, its *up* output is max(*a*,*b*) and the *down* output is min(*a*,*b*); for the unit *downsort*, its outputs are defined oppositely. As the sorting task has been identified as a computational bottleneck (refer to Fig. 3-1), symbols, schematics and function tables of the *upsort* and the *downsort* are shown in Figure 3-11, for possible ASIC realization of the module *culler*. Note that, since a negative ray/patch distance (when bit 31 in Fig. 3-10 is 1) means the ray runs away from the patch (refer to Section 2.7), it is always treated as "greater" than a positive distance and therefore will be eliminated by the *culler*. The logic expressions for the signal *upsel* are $upsel = \overline{a[31]}(b[31]+\overline{GT})$ and $upsel = \overline{b[31]}(a[31]+GT)$ respectively, in units *upsort* and *downsort*.



a[31]	b[31]	GT	upsort				downsort			
			upsel	downsel	up	down	upsel	downsel	up	down
0	0	0	1	0	b	a	0	1	a	b
0	0	1	0	1	a	b	1	0	b	a
0	1	x	1	0	b	a	0	1	a	b
1	0	x	0	1	a	b	1	0	b	a
1	1	x	0	1	a	b	0	1	a	b

Figure 3-11. Symbols, schematics and function tables for functional units *upsort* and *downsort*

Using the two functional units *upsort* and *downsort* as elements, a bitonic sorting network is constructed as shown in Fig.3-12. It takes 16 sorting keys (tagged ray/patch distances) as inputs and outputs the tags (the Patch Index in Fig. 3-10) of the four keys with the smallest ray/patch distance values. Sorting elements in the dashed blocks are not required since only four outputs are needed.

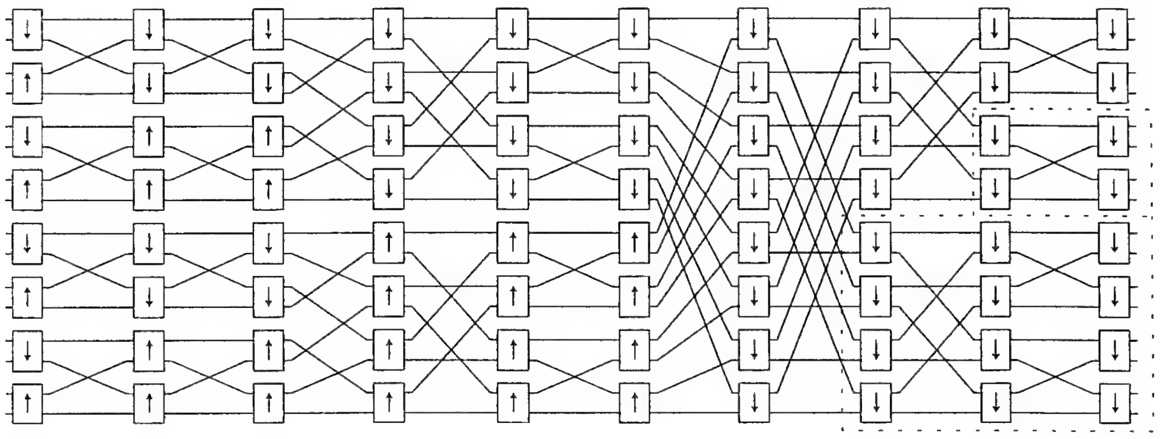


Figure 3-12. The sixteen -inputs bitonic sorting network

Based on the bitonic sorting network, module *Culler* can be completed and its block diagram is given in Fig. 3-13.

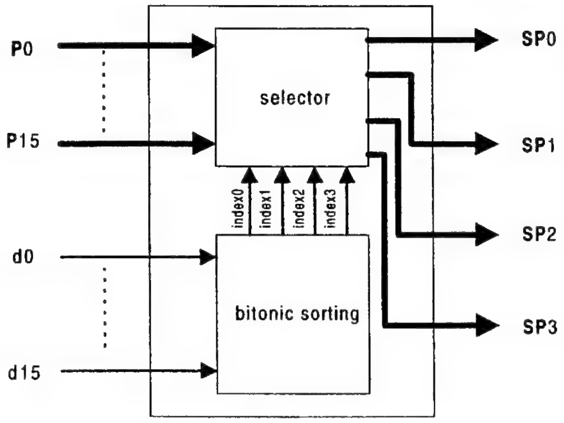


Figure 3-13. The Culler module

3.10 The System Architecture

Base on the load distribution among tasks of the proposed algorithm (refer to Fig. 3-1), and the functional building blocks described in the previous subsections, the overall system architecture for physical implementation of the proposed Ray/Bézier-patch intersection algorithm is given in Fig. 3-14.

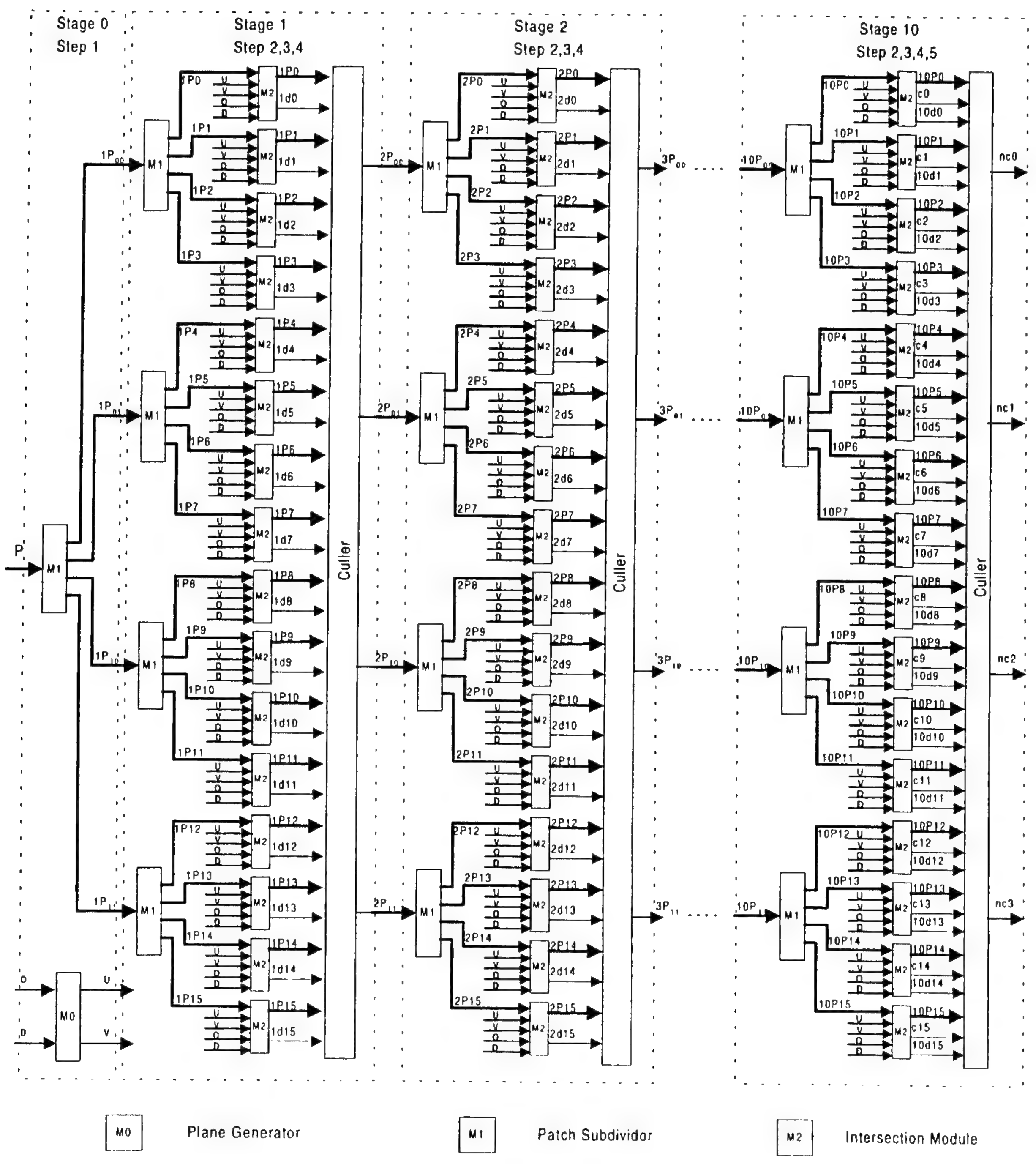


Figure 3-14. The overall system architecture

The proposed architecture contains eleven stages for pipelining. Stage 0 executes Step 1 of algorithm; Stage 1 to 10 execute the loop body of the algorithm. They are identical except that stage 10 outputs centers of the last four sub-patches that will be taken as the ray/patch intersection points. Within each stage, the inherent parallelism of the algorithm is explored and supported by the functional building blocks discussed in Subsections 3-1 through 3-9. The number of stages was determined based on observations of the algorithm's convergence rate during the software verification process. Throughput of the overall system will be determined by the latency of a single stage. Note that, the system architecture shown in Fig. 3-14 illustrates the datapath stream of the proposed algorithm with coarse-grained pipeline stages. It is open to refined intra-stage pipelining and hardware/software partitioning, based on specific performance requirements and implementation constraints determined by different applications.

4. Performance Analysis

Because of the complexity of the proposed algorithm, and the necessity of executing the algorithm in floating-point data format, performance evaluation based on implementation models containing significant amount of custom designed hardware will be beyond the scope of this study. Our evaluation results were obtained by assuming that, except for the bitornic sorting network (the identified computational bottleneck), an application-specific system for accelerating the algorithm will be built exclusively on the off-the-shelf TMS320C6701 digital signal processors (DSPs). We chose the TMS320C6701 because it is representative to the latest generation of high-performance DSPs, and its development tools are readily available. Although our analysis is based on a specific implementation model, it provides a good insight for future efforts in physical implementation.

4.1 Overview of the TMS320C67x Family of Digital Signal Processor

The TMS320C67x DSPs are the floating-point versions of the TMS320C6000 DSP family. The TMS320C6701 (a.k.a. 'C6701) DSP is based on the high-performance, advanced VelociTI very-long-instruction-word (VLIW) architecture developed by Texas Instruments (TI™), which made this device outperform the earlier DSPs and traditional superscalar designs through increased instruction-level parallelism. Block diagram of 'C6701's CPU core is given in Fig. 4-1. It has 32 general-purpose registers of 32-bit word length and eight highly independent functional units that can execute the functions summarized in Table 4-1, for both IEEE single- and double-precision floating-point instructions. It is able to execute up to eight 32-bit instructions or two multiply-accumulates (MACs) per cycle. Running at a clock rate of 167 MHz (6 ns cycle time), the 'C6701 DSP can deliver up to 1 giga floating-point operations per second (GFLOPS). Therefore, it possesses the computing power of high-speed array processors while maintaining the programming flexibility of regular microprocessors.

The 'C6701 includes a large bank of on-chip memory and has a powerful and diverse set of peripherals. Program memory consists of a 64K-byte block which can be user configurred as cache or memory-mapped program space. Data memory consists of two 32K-byte blocks of RAM. The peripheral set includes two multichannel buffered serial ports (McBSPs), two general-purpose timers, a host-port interface (HPI), and a glueless external memory interface (EMIF) capable of interfacing to SDRAM or SBSRAM and asynchronous peripherals.

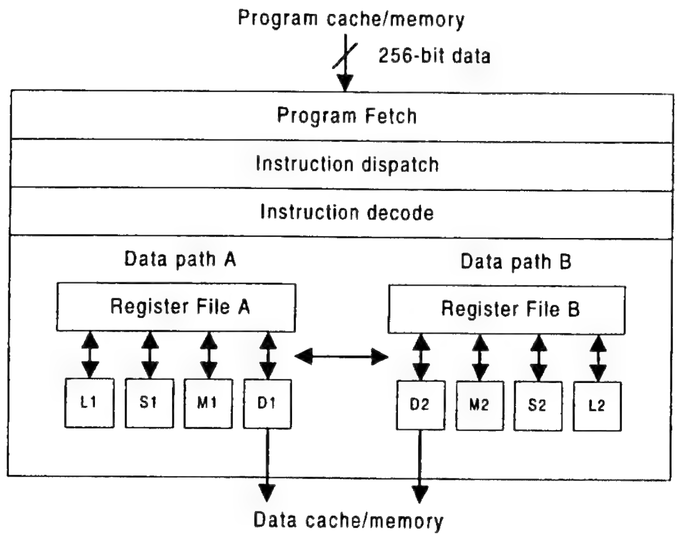


Figure 4-1. The C67x CPU core

Functional Unit	Fixed-Point Operations	Floating-Point Operations
L unit(L1, L2)	32/40 arithmetic and compare operations 32-bit logical operations	Arithmetic operations conversion operations
S unit(S1, S2)	32-bit arithmetic operations 32/40-bit shifts and 32-bit bit-field operations 32-bit logical operations Branches Constant generation	Compare Absolute value operations
M unit(M1, M2)	16x16 bit multiply operations	32x32 bit fixed-point multiply operations Floating-point multiply operations
D unit(D1, D2)	32-bit add, subtract, linear and circular address calculation Loads and stores	Load double word

Table 4-1. 'C6701 Functional Units
(Fixed-point operations and floating-point operations are both available on 'C6701)

4.2 Techniques for Optimizing the TMS3206701 Coding

The TMS3206701 is supported by a complete set of software development tools which include: an optimizing C compiler, an assembly optimizer, an assembler, a linker, assorted utilities, and a Windows^{TM} debugger interface for visibility into source code execution. In this study, we have used the following tools/techniques to optimize the 'C6701 coding:

Data Dependency Detection

To maximize the efficiency of the source code, the 'C6701 compiler schedules as many instructions as possible in parallel. In order to do so the compiler must determine dependencies between instructions. The following techniques have been used to help the compiler determine instructions that are independent to each other:

- Use the *const* keyword to indicate which objects are not changed by a function

- Use the *-pm* (program-level optimization) option, which gives the compiler global access to the whole program or module and allows it to be more aggressive in ruling out dependencies
- Use the *-mt* option, which allows the compiler to use assumptions that allow it to eliminate dependencies.

Software pipelining

Software pipelining is a technique used to schedule instructions from a loop so that multiple iterations of the loop execute in parallel. When the *-o2* or *-o3* option is invoked, compiler attempts to software pipeline the code. Fig. 4-2 shows the concept of software pipelining.

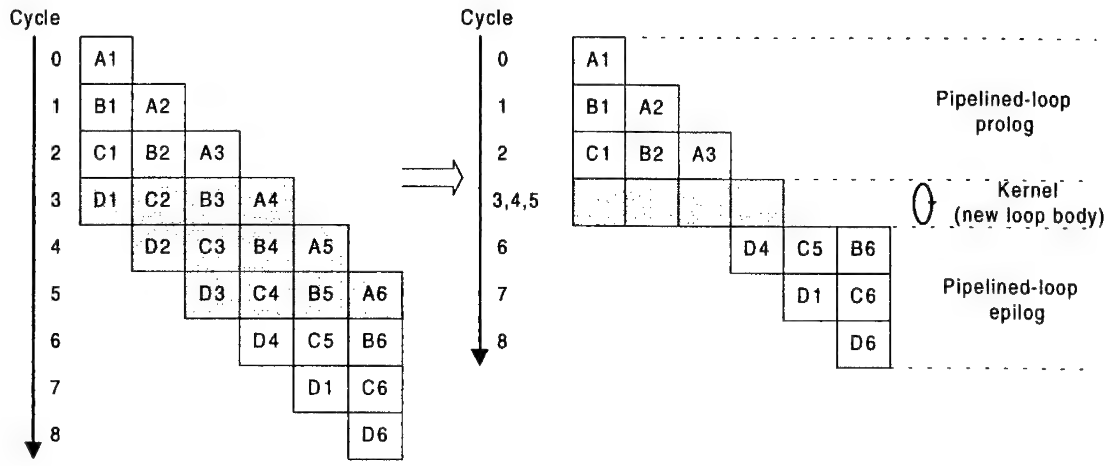


Figure 4-2. Software pipelining

In Fig. 4-2 the stages of the loop are represented by A, B, C and D. The figure shows that a maximum of six iterations of the loop can execute at one time. The shaded area represents the repeated pattern known as the *kernel*. The area immediately before and after the kernel is called the pipelined-loop *prolog* and *epilog*, respectively. In software pipelining, the original loop is essentially replaced by the kernel which becomes the body of a new loop where all four stages execute in parallel. The kernel determines the efficiency of software execution.

A *trip-count* is the number of times that a loop executes. Software pipelining is invoked by passing the trip-count to the compiler with the following methods:

- use the *_nassert* intrinsic to convey the trip count to compiler

```
_nassert (x==40)
for (j = 40; j !=0;  j--)
```

- count down the trip counter

```
Before    for ( i=0; i < N; i++)  /* N = trip-count*/
After     for (i = N; i !=0; i--)  /* N = trip-count*/
```

- use the *-ms0* option to instruct the compiler to only generate the software-pipelined code when the trip count is known

Under certain conditions, a loop cannot be software-pipelined. See References [SPRU198C] and [SPRU301A].

Loop unrolling

Another technique that improves performance is unrolling the loop, which means, expanding small loops so that each iteration of the loop appears in the code. The optimization increases the number of instructions available to execute the loops in parallel. Loop unrolling is used when the operations in a single iteration do not use all of the resources of the C6701 architecture.

Speculative execution

The *-mh* option eliminates the epilog for a software pipelined loop, which can result in significant savings in code size. Software pipelined loop epilogs can often be eliminated if load instructions can be speculatively executed before knowing whether the results of the executions are needed.

4.3 Code Optimization and Function Call Profiling

With the TMS320C6701 development tool set *Code Composer Studio*, all function calls listed in Table 3-1 are optimized and profiled. During the optimization process, the 'C6701's C-compiler has fully considered the delays from load, store and the branching, and the code generated from C source code took full advantage of the software pipelining technique and the 'C6701's superscalar architecture. When profiling the function calls, all data are declared as single precision floating-point numbers. As the stand-alone simulator does not take into consideration about the memory bank conflicts (if the memory uses interleaved banks), the external memory access stalls (if the external memory is not ready), and the existence of cache in the memory hierarchy (which will make the programs run faster), all data storage were kept word aligned, thus eliminating the memory bank conflicts. We took multiple steps to optimize a function call. Each time a new optimization technique is applied based on the code obtained from the previous step. The function call *PatchSubdividor* is used as an example to show how effective the optimization techniques are. Fig. 4-3 shows the function's 'C6701 runtime after each optimization step; and Table 4-2 lists the optimization techniques applied in each of the steps.

Source codes for all the other function calls are optimized in a similar fashion. Runtimes of the optimized function calls on 'C6701 are shown in Table 4-3.

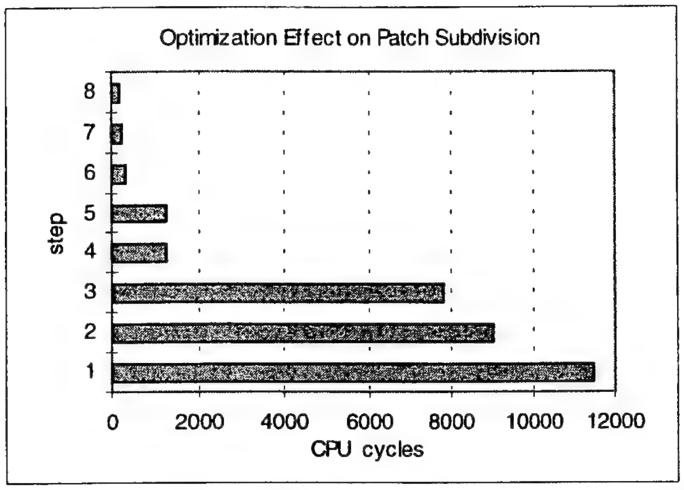


Figure 4-3. The effect of code optimization on the function call *PatchSubdivisior*

Table 4-2: Step-by-step optimization procedures on the function call *PatchSubdividor*

Step	'C6701 CPU cycles	Optimization Methods Applied
1	11478	none
2	9018	declare parameters as *const* type
3	7803	file-level optimization by compiler
4	1284	invoke software-pipelining ; eliminate divide routine call by replacing multiply
5	1257	loop unrolling
6	317	force constant in expression to float
7	243	speculative execution

Table 4-3. Runtime of optimized function calls on the 'C6701 DSP

Name of Function Call	'C6701 CPU Cycles	Comments
Split-in-half	182	called 3 times by *PatchSubdividor*
GetPatchCenter	102	
PlaneGenerator	36	
PatchXPlane	112	hull classification on one plane
GetDistance	28	
Quicksort	3081	

4.4 An ASIC Solution to the Bitonic Sorting Network

From Table 4-3, it is clear that the function call *QuickSort* optimized for the 'C6701 is still the computational bottleneck of the proposed algorithm. Consequently, a hardware solution to the problem should be considered. Recall that in Section 3.9 we defined a 10-stage bitonic sorting network for parallel execution of the *quick sort* algorithm (refer to Fig. 3-12); schematics of the two basic functional units *upsort* and *downsort* for constructing the sorting network were also given. In order to obtain some actual timing information, we implemented the *upsort* and *downsort* on Xilinx XC4025E-3 FPGAs. Timing simulations on these functional units showed that their maximal delay time is less than 80 ns. The overall sorting network was than verified via VHDL simulation. The estimated overall delay of the 10-stage sorting network is 800 ns without any intra-stage pipelining. The estimated hardware resource is 3,392 CLBs on the XC4025E devices.

4.5 Performance Esetimation

Using the overall system architecture proposed in Section 3.10 (refer to Fig. 3-14) as the architectural framework, there are many options in implementing the physical system, depending on resource constraints and performance requirements on specific applications. The system architecture shown in Fig. 3-14 contains ten identical coarse-grained pipeline stages (excluding Stage 0 which is for preprocessing), each stage can be divided into sub-stages for refined pipelining, by load balancing (static) the tasks within a single stage, based on the timing information discussed in Sections 4.3 and 4.4. Two specific implementation cases are discussed here and for each implementation, performance estimation will be given.

In the first implementation, each stage in Fig. 3-14 is divided into four sub-stages each containing a single 'C6701 DSP. The original Stage 0 (for preprocessing) is merged into the Stage 1. Load balancing is then applied to the computing tasks within Stage 1 followed by re-profiling. The detailed information on Stage 1 is given in Table 4-4, where the performance estimation is based on the execution time required Sub-stage 2 as it has the largest latency.

Table 4-4. Performance estimation based on Implementation Case 1

Sub-stages in Stage 1	Number of 'C6701s used in each sub-stage	Function call(s) executed by each 'C6701	Number of 'C6701 clock cycles used (6 ns/cycle)	Pipeline Cycle in seconds	Overall system throughput in RPI/s (ray-patch intersection per second)
1	1	*Split_in_half* (called 15 times) *PlaneGenerator* (called 1 time)	2803	2.15×10^{-5}	4.65×10^{4}
2	1	*PatchXPlane* (called 32 times)	3584		
3	1	*GetPatchCenter* (called 16 times) *GetDistance* (called 16 times)	3328		
4	1	*Quicksort* (called 1 time)	3081		

In the second implementation, Stage 0 in Fig. 3-14 needs to be implemented separately and each of the rest stages is divided into five sub-stages each containing multiple 'C6701 DSPs except the Sub-stage 5 (the *culler*), which is implemented in hardware (refer to Section 4.4). Again, static load balancing is then applied to the computing tasks within Stage 1 followed by re-profiling. The refined pipeline stages are illustrated in Fig. 4-4, and the detailed information on Stage 1 is given in Table 4-5, where the performance estimation is based on the execution time required Sub-stages 1 and 2 as they have the largest latency.

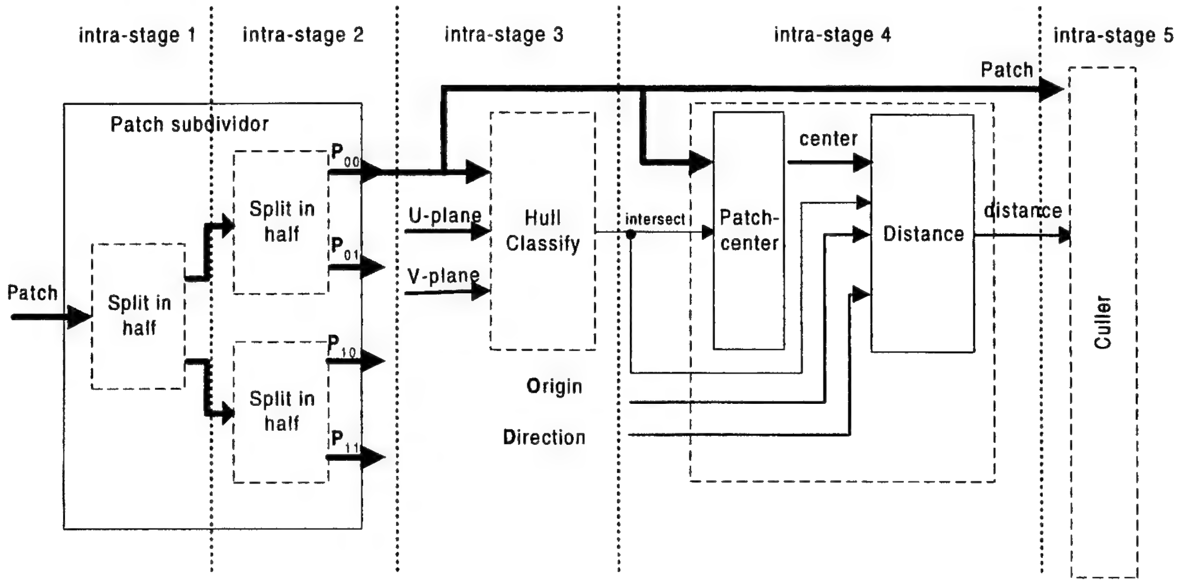


Figure 4-4. Sub-stages inside a typical stage in Implementation Case 2

Sub-stages in Stage 1	Number of 'C6701s used in each sub-stage	Function calls executed by each 'C6701	Number of 'C6701 clock cycles used (6 ns/cycle)	Pipeline Cycle in seconds	Overall system throughput in RPI/s (ray-patch intersection per second)
1	4	*Split_in_half* (called 1time)	182	1.092×10^{-6}	9.16×10^{5}
2	8	*Split_in_half* (called 1time)	182		
3	32	PatchXPlane (called 1 time)	112		
4	16	*GetPatchCenter* (called 1 time) GetDistance (called 1 time)	130		
5	None (by ASIC)	*Quicksort* (executed in hardware)	N. A. (800 ns delay)		

Table 4-5. Performance estimation based on Implementation Case 2

This implementation leaves a lot of room for further expedition of the execution. For instance, optimizing the code for the function *Split_in_half* at assembly language level could reduce the number of '6701 cycles required by Sub-stages 1 and 2 and thus the pipeline cycle time could be shortened. The two case studies show that: 1) for more significant performance gain, it would be more efficient to fine-grain the pipeline stages and fully support the inherent parallelism for each computing tasks inside the stages; 2) when the pipeline stages are finer grained, the ASIC approach would be a more cost-effective option for implementation.

5. Conclusion

In this study, an algorithm for solving the ray/Bézier-patch intersection problem has been developed and verified. Although its computational complexity is on the same order of some known algorithms, the proposed algorithm has the advantage of being suitable for parallel and pipelined execution. The corresponding systems architecture for physical realization has also been developed with all the functional building blocks specified. Computing tasks required by the proposed algorithm are profiled on the TMS320C6701 DSP and the computational bottleneck is identified. To eliminate the bottleneck, a hardware solution has been proposed and verified. Performance data are obtained based on two implementation models built upon the 'C6701 DSPs, with and without additional custom hardware, respectively. The data show that with the proposed algorithm, more than one million RPIs/s (ray-patch intersections per second) could be calculated on specially designed computing systems based on the proposed architecture, with the pipeline stages being refined and the inherent parallelism of computing tasks inside the pipeline stages being supported. The data also suggest that higher performance can be achieved by optimizing the code at assembly level (for functional blocks implemented in software) or more efficiently, by replacing the functional building blocks with custom-designed hardware. We anticipate that with the rapid technological advances, some practical systems based on this study could be built in the near future.

Acknowledgments

The author wishes to acknowledge the financial support given by the AFOSR for making this work possible. The author also would like to acknowledge and thank the following individuals, for their participation, assistance and encouragement:

At Washington State University:	Dr. Robert Lewis, graduate students Renwei Wang and Shuyan Li;
At the AFRL/IFSD Wright Research Site:	my team leader Luis Concha, teammates Brian Kadrovach, Britton Read, Keith Pedersen, David Bawcom and many friends;
At the RDL:	Ms. Johnetta Thompson and Ms. Rebecca Kelly-Clemennons.

Without them, this project would never have been completed.

References:

[Hai89] Eric Haines, " Essential ray tracing algorithms," Chapter 2 in *An Introduction to Ray Tracing*, A.S. Glassner Ed., Academic Press, 1989.

[AK89] James arvo, David Kirk, "A survey of ray tracing acceleration techniques",Chapter 6 in *An Introduction to Ray Tracing*, A.S. Glassner Ed., Academic Press, 1989.

[Hung98] Donald L. Hung, " *A Study on Accelerating the Ray/Triangular-facet Intersection computation in Xpatch*", Technical Report to the U.S. Air Force, August 1998.

[FP79] I.D.Faux, MJ.Pratt, "*Computational Geometry for Design and Manufacture*", Ellis Horwood Ltd., 1979

[FB94] Alain Fournier and John Buchanan, "*Chebyshev Polynomials for Boxing and Intersections of Parametric Curves and Surfaces*", EUROGRAPHICS'94, Vol. 13,No 3, pp129-142.

[KAJ82] James T. Kajiya, "*Ray tracing Parametric Patches*", Computer Graphics, 16(3) pp245-254, July 1982

[BS93] W.Barth and w.Sturzlinger, "*Efficient Ray Tracing for Bézier and B-spline Surfaces*", Computer Graphics Vol.17,No 4, pp423-430,1993

[CSS97] Swen Campagna, Philipp Slusallek, Hans-Peter Seidel, "*Ray tracing of sline surfaces: Bézier clipping, Chebyshev Boxing, and bounding volume hierarchy - critical comparison with new results*", The Visual Computer,Vol.13,pp265-282,1997

[NSK90] Tomoyuki Nishita, Thomas W. Sederber, Masanori Kakimoto, "*Ray tracing trimmed rational surface patches*", Computer Graphics, Vol24, No 4, pp337-345, Aug.1990

[KK96] Hyun-Joon Kim,Chong-Min Kyung, "*A new parallel ray-tracing system based on object decomposition*", The Visual Computer,Vol.12, pp244-253, 1996

[BB94] Didier Badouel, Kadi Bouatouch, "*Distributing data and control for ray tracing in parallel*",IEEE Computer Graphics and Application,Vol.14, No.4, pp69-77,July 1994

[FP92] Alian fournier, Pierre Poulin, "*A Ray Tracing Accelerator based on a Hierarchy of 1D Sorted List*", Technical Report:UBC CS TR-92-37, http://www.cs.ubc.ca/cgi-bin/tr/1992/TR-92-37

[FOL97] James D. Foley... "*Computer Graphics Principles and Practice*",2nd in C, Addison Wesley, 1997

[WW98] Alan Watt, Mark Watt, "*Advanced Animation and Rendering Techniques Thory and Pratice*", Addison Wesley 1998

[CA84] Joseph J.F. Cavanagh, "*Digital Computer Arithmetic: Design and Implementation*", 1984, McGraw-Hill

[SPRU187E] "*TMS320C6000 Optimizing C Compiler User's Guide*", Texas Instruments, 1999

[SPRU189D] "*TMS320C6000 CPU and Instruction Set Reference Guide*", Texas Instruments, 1999

[SPRU198C] "*TMS320C62x/67x Programmer's Guide*", Texas Instruments, 1999

[SPRU301A] "*TMS320C6000 Code Composer Studioo Tutorial*", Texas Instruments, 1999

[PH98] David A.Patterson and John L. Hennessy, "*Computer Organiztion & Design:the Hardware/software interface*", Morgan Kaufmann, 1998

[SFK97] Dezso Sima, Terence Fountain, Péter kacsuk, "*Advanced Computer Architectures: a Design Space Approach*", Addison Wesley, 1997

TRANSFORM METHODS FOR WATERMARKING DIGITAL IMAGES

Adam Lutoborski
Professor of Mathematics
Department of Mathematics

Syracuse University
Syracuse, NY 13214-1150

Final Report for:
Summer Research Extension Program
Air Force Research Laboratory, Rome NY

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, Washington, D.C.
and
Syracuse University

December 1999

Transform Methods for Watermarking Digital Images

Adam Lutoborski
Professor of Mathematics
Department of Mathematics
Syracuse University

December 14, 1999

Abstract

We investigate a class of wavelet based transform methods for watermarking digital images. An transform method consists of a discrete wavelet transform followed by iterated transforms optionally and a random additive transform. We analyze DWT-projection transform and two DWT-SVD multiplicative transforms DWT-SVD-SV and DWT-SVD-D.

TRANSFORM METHODS FOR WATERMARKING DIGITAL IMAGES

Adam Lutoborski

1 Previous Results and Project Motivations

Digital watermarking is embedding of imperceptible markings into digital images (and other signals). The watermark depends on the owner's secret numerical key and can be detected by using an appropriate numerical detection algorithm. The crucial property of the watermark is its ability to withstand (to the similar degree as the image itself) image modifications resulting from an assortment of image processing operations such as compression and filterings of various kinds.

Watermarking techniques for digital images can be divided into two groups: spatial domain methods which insert the watermark directly in the pixel values of the image and transform methods which insert the watermark in the output of a certain transform of the image.

Many important general aspects of the transform based watermarking methods have been thoroughly discussed in the pioneering papers of I. Fridrich [3], [4], [5], [6], [7], A.H. Tewfik [18], [14], [15], [16] and other scientists working in the general area of signal processing and specifically image processing.

Most of the transform based watermarking techniques are based on the discrete Fourier transform (DFT) and most often on the discrete cosine transform (DCT), see [13], [14]. A sophisticated transform watermarking method based on the DCT was developed by J. Fridrich in [7]. Two independent watermarks are inserted in the DCT coefficients of the input image. To produce a low frequency watermark, N lowest frequency DCT coefficients are modified in a controlled way using specially introduced index function. Next a spread spectrum technique is used to modify the middle frequency DCT coefficients. The resulting superposition of the two watermarks guarantees high robustness with respect to JPEG compression, mosaic filter and pixel permutation but also to contrast / brightness adjustments.

In their report "On Digital Watermarks" J. Fridrich, A. Baldoza and R. Simard [7] emphasized the importance of using the transform methods in obtaining robust watermarks in digital images. In the conclusions the authors made several important observations concerning the security (ability to withstand an attack) of their DCT based watermarking method. They note that "methods in which watermarks are spanned by publicly known basis functions may provide less security than previously thought". It is in response to the quoted concern that the wavelet transform based methods are beginning to appear recently in the literature. We refer here the work of D. Kundur and D. Hatzinakos [8], [9], [1], [2], [20], [21] and the investigator in [10] .

2 General Transform Methods in Watermarking

It is necessary to introduce some of the basic notations and definitions to enable us to present our previous results and formulate the results of this project.

Digital images, denoted by capital Latin letters, will be matrices with integer entries in the interval $[0, 255]$. The set of all such images is denoted by $\mathbb{I}$. The inner product of matrices $A, B \in \mathbb{R}^{n\times n}$ is given by $\langle A, B\rangle = \text{trace}(A^T B)$ and the Frobenius norm is given by $||A|| = \langle A, A\rangle^{1/2}$. The Frobenius norm is very useful in matrix computations but $||A - B||$ is known to be an inadequate tool for measuring perceptual distance between images A and B. It is an open question in HVS (human visual system) how to define a distance function $\rho : \mathbb{I} \times \mathbb{I} \to \mathbb{R}$ such that $\rho(A, B)$ is an appropriate "visual" distance between images A and B.

An algorithm which for a given input image M and a value of a key parameter k, $k \in \mathbb{N}$ constructs a watermarked image $\mathcal{W}(M)$ is called a watermarking algorithm. Any such algorithm defines a watermarking mapping

$$\mathcal{W} : \mathbb{I} \times \mathbb{N} \to \mathbb{I},$$

$$(M, k) \to \mathcal{W}(M, k).$$

The image $\mathcal{W}(M, k)$ is called the watermarked image and $\mathcal{W}(M, k) - M$ is called the image watermark of M. The output of the watermarking mapping $\mathcal{W}(M, k)$ depends on two variables: the original image M and the value k of a secret key for a random number generator which will supply a random parameter to be used in determining the watermark. It is assumed that the dependence $\mathcal{W}(\cdot, k)$ is continuous and is known (and subject to attack) but the dependence $\mathcal{W}(M, \cdot)$ being random is not known.

We say that the watermark $\mathcal{W}(M, k) - M$ is detectable in the image N with correlation

τ, $0 \leq \tau \leq 1$ if and only if

$$C_M(\mathcal{W}(M,k),N) = \frac{|\langle \mathcal{W}(M,k) - M, N - M\rangle|}{||\mathcal{W}(M,k) - M|| ||N - M||} \geq \tau.$$

If the above value of τ is large and close to 1 the probability that $N = \mathcal{W}(M,k)$ is high. If

$$C_M(\mathcal{W}(M,k), \mathcal{W}(M,m)) \leq \tau_{\mathcal{W}} \qquad \text{for all} \quad m \neq k, \quad m \in \mathbb{N}$$

then we say that a detection threshold for the watermark $\mathcal{W}(M,k) - M$ is $\tau_{\mathcal{W}}$. A detection threshold must be set for the above detection procedure before testing any watermarking method for robustness. To assure the effectiveness of the detection procedure the threshold must minimize the number of false detections and non-detections.

The detection function $C_M(\mathcal{W}(M,k),N)$ requires the explicit knowledge of the original image M. All such detection functions are called non-oblivious. The evaluation of the detection function C_M is disproportionately cheap in comparison with the evaluation of the watermarking mapping. Not surprisingly, it provides only a very basic statistical correlation measure between two sets of data. The detection function C_M is completely unrelated to the important data characteristics. The more $\mathcal{W}(M,k)$ depends on the image M, as might be necessary, the closer the correlation $C_M(\mathcal{W}(M,k), \mathcal{W}(M,l))$ will be to the detection threshold $\tau_{\mathcal{W}}$ making detection impossible.

In view of these observations we may formulate some basic requirements for the watermarking mapping and more specifically for the dependence $\mathcal{W}(M,\cdot)$. Assuming that M, N are given and $\mathcal{W}(M,\cdot)$ is known, we can identify N as $\mathcal{W}(M,k)$ only if we know the secret key $k \in \mathbb{N}$ or if we know $k \in \mathbb{N}$ and $\mathcal{W}(\tilde{M},k)$ where $\tilde{M}$ is very close to M.

We can only identify N if

$$C_M(\mathcal{W}(M,k),N) \leq \tau_{\mathcal{W}}.$$

Therefore the dependence $\mathcal{W}(M,\cdot)$ should be such that $\mathcal{W}(M,k)$ and $\mathcal{W}(M,l)$ are both as far from M as it is "visibly" possible and at the same time the inner product of these matrices should be as small as possible for $k \neq l$. The ideal (for the detection function C_M) watermarking mapping $\mathcal{W}(M,\cdot)$ should have the property that $C_M(\mathcal{W}(M,k), \mathcal{W}(M,l)) = 0$ for $k \neq l$. Figure 1 illustrates such requirement.

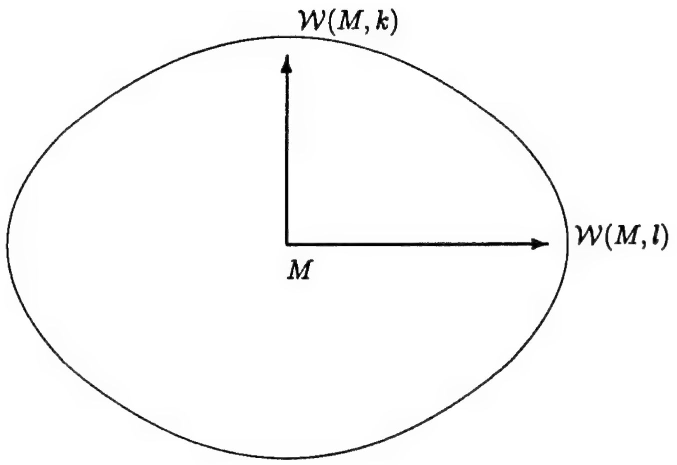


Figure 1: Ideal watermarking mapping $\mathcal{W}(M,\cdot)$

3 Discrete Wavelet Transforms and Digital Watermarks

One of the most promising and effective transform techniques in modern signal processing is the wavelet transform which is computed by correlating a given signal with dilates and translates of a single function ψ possessing exceptional time and frequency localization properties. A wavelet ψ is a function (square integrable with mean zero) such that the family of its dilates and translates

$$\left\{2^{\frac{2}{j}}\psi(2^j t - k)\right\}_{j,k\in\mathbf{Z}}$$

is an orthonormal basis in the space $L^2(\mathbb{R})$.

For any square integrable function f we have the following expansion in the above wavelet basis

$$f(t) = \sum_{j\in\mathbf{Z}}\sum_{k\in\mathbf{Z}} d_{j,k}\psi(2^j t - k)$$

with

$$d_{j,k} = 2^{\frac{j}{2}}\int_{-\infty}^{\infty} f(t)\psi(2^j t - k)\,dt.$$

The set $\{d_{j,k}\}_{j,k\in\mathbf{Z}}$ of expansion coefficients is called the discrete wavelet transform (DWT) of the signal f. We may interpret the partial sums in the above expansion with respect to j as the approximations to f at the resolution 2^j. Formally the approximation to f at the resolution 2^j is defined as an orthogonal projection of f onto a space $V_j \subset L^2(\mathbb{R})$. The ascending sequence $\{V_j\}_{\mathbf{Z}}$ of such spaces is called a multiresolution approximation of $L^2(\mathbb{R})$ when each V_j is spanned by dilates and translates of a single function ϕ called a scaling function.

Using both the wavelet and the scaling function we may rewrite the expansion for f as:

$$f(t) = \sum_{k \in \mathbf{Z}} 2^{\frac{j}{2}} a_{j,k} \phi(2^j t - k) + \sum_{j=J}^{\infty} \sum_{k \in \mathbf{Z}} 2^{\frac{j}{2}} d_{j,k} \psi(2^j t - k).$$

The expansion coefficients $a_{j,k}$ are called approximation coefficients and the coefficients $d_{j,k}$ are called the detail coefficients at the resolution 2^j or level j. The above decomposition can be computed recursively for j by the fast dyadic algorithm of Mallat [11].

The approximation coefficients are given by:

$$a_{j-1,k} = \sum_{l \in \mathbf{Z}} h_{l-2k} a_{j,l}$$

and the detail coefficients are given by:

$$d_{j-1,k} = \sum_{l \in \mathbf{Z}} g_{l-2k} a_{j,l}$$

where the coefficients h_n, g_n depend only on the scale j on l and the scaling function ϕ.

The coefficients $a_{j,k}$ can be reconstructed from the approximation and detail coefficients at the next coarser scale from:

$$a_{j,k} = \sum_{l \in \mathbf{Z}} h_{l-2k} a_{j-1,l} + \sum_{l \in \mathbf{Z}} g_{l-2k} d_{j-1,l}.$$

Unlike the DCT the DWT is not a single vector (or a matrix in the case of images). The sets of the approximation and detail coefficients at all levels form a binary tree of coefficients. The DWT has a recursive structure at each step requiring computing convolution and decimation (quadrature filter), see [19] for comprehensive treatment of wavelet programming technology.

Image processing deals with discrete, finitely supported functions of 2 variables and in order to put to use the one dimensional wavelet transform two significant steps are to be taken. First, one dimensional wavelet base has to be modified for use on a finite interval by using periodic, folded or boundary adapted wavelets. Next, one dimensional wavelet algorithms have to be extended to two dimensions by taking separable approach and working with tensor product formulas.

Expanding a function of one variable over a finite interval requires modifying the wavelet bases. This can be achieved by using periodic, folded or boundary adapted wavelets. Assume that the approximation and detail coefficients at the resolution J are finite and periodic outside a finite interval and given by vectors a^J, d^J, $a^J = (a_{J,m})_{m=0,\dots,2^J-1}$, $d^J = (d_{J,m})_{m=0,\dots,2^J-1}$.

We can formulate the dyadic DWT algorithm for computing approximation and detail coefficients at the next coarser level in the matrix form as computing the vectors $a^j = (a_{j,m})_{m=0,\dots,2^j-1}, d^j = (d_{j,m})_{m=0,\dots,2^j-1}$ for $j = 2^j, \dots, 1$ from the following formulas

$$a^{j-1} = H_{j-1}a^j$$
$$d^{j-1} = G_{j-1}a^j$$

and the inverse dyadic DWT from

$$a^j = H_{j-1}^T a^{j-1} + G_{j-1}^T d^{j-1}$$

where H_j, G_j are $2^{j-1} \times 2^j$ matrices with entries $(H_j)_{k,l} = h_{l-2k}$, $(G_j)_{k,l} = g_{l-2k}$.

Using the above notation the DWT in matrix terms can be formulated as an orthogonal transformation Q of the input vector a^J into vector b^J containing the coarse approximation a^0 and all intermediate detail coefficients

$$Qa^J = \begin{pmatrix} d^{J-1} \\ \vdots \\ d^0 \\ a^0 \end{pmatrix} = b^J.$$

The matrix Q is called the DWT matrix. The product Qa^J can be computed efficiently at a cost of $\mathcal{O}(N)$ where $N = 2^J$ which is asymptotically faster than the DCT transform requiring $\mathcal{O}(N \log_2 N)$ operations.

In the case of a square $N \times N$ image $M = A^{(J)}$, $N = 2^J$ the separable DWT using wavelet ψ results in $3J + 1$ submatrices

$$B^{(J)} = \left(A^{(0)}, \{D_1^{(j)}, D_2^{(j)}, D_3^{(j)}\}_{j=J-1,\dots,0}\right)$$

see [11]. Introducing a corresponding DWT transformation we may write it as

$$\mathcal{F}_\psi(A^{(J)}) = B^{(J)}.$$

The approximation $A^{(j-1)}$ at the next coarser level is obtained from $A^{(j)}$ by one step of the dyadic DWT algorithm. The resulting transform is denoted by

$$\mathcal{F}_{\psi,j}(A^{(j)}) = \left(A^{(j-1)}, \{D_1^{(j-1)}, D_2^{(j-1)}, D_3^{(j-1)}\}\right),$$

and if the context doesn't require it we will abbreviate $\mathcal{F}_{\psi,j} = \mathcal{F}_j$.

Consequently, by composing J steps of dyadic algorithm we obtain $A^{(0)}$ the coarse approximation of the input image $A^{(J)}$:

$$\mathcal{F}_1 \dots \mathcal{F}_J(A^{(J)}) = A^{(0)}.$$

Our description of the rudiments of the DWT transform is now complete and using it we can concisely present two DWT-based watermarking methods developed by the investigator in [10].

Both our transform-based watermarking methods consist of three main steps: wavelet transform, watermarking transformation and the inverse wavelet transform.

Analytically, let $A = A^{(J)}$ and $1 \leq l \leq J$. First, l steps of the dyadic DWT are computed:

$$\mathcal{F}_{J-l+1} \dots \mathcal{F}_J(A^{(J)}) = \left(A^{(j-l)}, \{D_1^{(J-l)}, D_2^{(J-l)}, D_3^{(J-l)}\}\right).$$

Next, the watermarking mapping $\mathcal{T}$ is applied to the approximation coefficients $A^{(J-l)}$

$$A^{(J-l)} \to \mathcal{T}(A^{(J-l)}).$$

Finally the watermarked image $\mathcal{W}(A)$ is reassembled by computing the inverse dyadic DWT:

$$\mathcal{W}(A) = \mathcal{F}_J^{-1} \dots \mathcal{F}_{J-l+1}^{-1}\left(\mathcal{T}(A^{(J-l)}, B^{(J-l)})\right).$$

To shorten this watermarking formula we rewrite it as:

$$\mathcal{W}(A) = \mathcal{F}_{\psi,J,l}^{-1}\left(\mathcal{T}(A^{(J-l)}, B^{(J-l)})\right).$$

In [10] we introduced and tested two additive watermarking mappings. The DWT-SVD watermark was defined as follows:

Let $A^{(J-l)} = U\Sigma V^T$ be the singular value decomposition of the transformed image. We set

$$\mathcal{T}(A^{(J-l)}) = U\Sigma V^T + \tilde{U}\bar{\Sigma}\tilde{V}^T$$

where $\bar{\Sigma} = \sigma$ diag(rand) is a random diagonal matrix with norm controlled by σ and $\tilde{U}$, $\tilde{V}$ are partially random modifications of orthogonal matrices U and V.

The Wavelet-random modulation watermark was defined as follows:

$$\mathcal{T}(A^{(J-l)}) = A^{(J-l)} + R$$

where R is a random, additive modulation obtained in a sequence of steps containing: (a) generation of a random matrix, (b) using cellular automaton to coalesce the pattern of the entries, (c) smoothing, (d) scaling. The above procedure was originally implemented by I. Fridrich [3] in a DCT-based watermarking algorithm.

The Wavelet-SVD watermark allows for partial analytical control of the magnitude, randomness, image dependency and watermarking selected parts of the space-frequency content of the image. Our numerical tests indicated that the Wavelet-SVD detection thresholds were lower, allowing easier detection and its was faster to embed. The SVD watermark embedded in $A^{(J-l)}$ could be detected at coarser levels in $A^{(J-k)}, k \geq l$ whereas random modulations could not be detected.

None of the two watermarking methods equipped with the same detection procedure (however each with its own threshold value) produced justifiably superior robustness results.

4 Iterated Transforms Methods in Watermarking

We begin by explaining some general reasons for using transform based watermarking. Computing a transform of an image is equivalent, in terms of linear algebra, to representing the image as a linear combination of vectors of a new basis. The coefficients of this linear combination constitute the transformation coefficients of the image. Transformation coefficients (as is the case for DCT and DWT) depend on a majority of the image pixels (image coefficients in the spatial basis). As a result the perturbations of individual transform coefficients affect almost all pixels of the image, or in other words, local changes of transform coefficients produce global changes in the image. The degree to which selective postprocessing of the transform coefficients affects the image depends strongly on the transform and can be best assessed analytically for the wavelet transforms.

Watermarking methods using DWT combined with iterated transforms has two underlying advantages:

- protection of the watermark from filtering operations which act on the image directly but locally

- making the detection procedures indirect and difficult by composing the primary transform with iterated key dependent crypto-transforms

Before discussing specific techniques it is necessary to list the main reasons for our interest in the wavelet transform based approach to watermarking:

- concision in representation of the transformed image
- robustness to wavelet-based compression algorithms
- locality in space-frequency allowing selective local transformations of the image and adapting the watermark to the image
- security resulting from wavelet transform selection, depth of transform selection and possibilities for design of key dependent crypto-wavelets
- computational cost proportional to the size of the input data

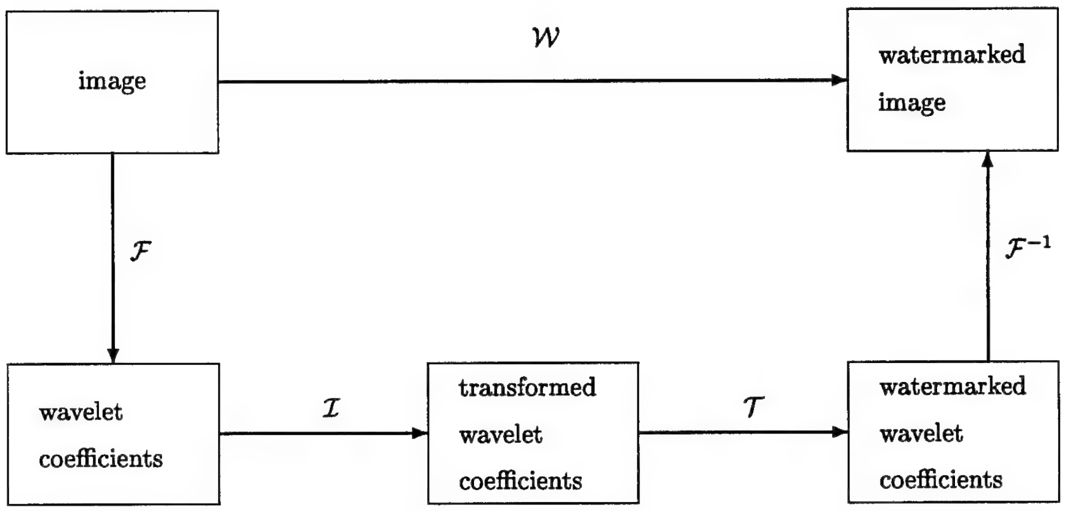


Figure 2: Iterated Transform Watermark

The iterated transform watermarking method consists of two basic steps: an iterated transform $\mathcal{I}$ and, optionally, a random additive transform $\mathcal{T}$. In this project we investigate only the iterated transforms, whereas the random additive transforms were studied in [10]. The composition of both these steps constitutes the watermarking transformation of the wavelet coefficients. We will describe in detail two basic types of transforms which we used in the transform $\mathcal{I}$: DWT-projection transforms and DWT-SVD multiplicative transforms. Both of these transforms can be applied separately or composed iteratively, entrywise and blockwise.

The roles of the components $\mathcal{I}$ and $\mathcal{T}$ in the watermarking method are different. Iterated transform $\mathcal{I}$ preserves the norm of the wavelet coefficients and is used to continuously and selectively modify subsets of wavelet coefficients. We interpret $\mathcal{I}(A)$ as a DWT corresponding to an unspecified crypto-wavelet whose properties depend on the user-defined secret parameter on which $\mathcal{I}$ depends. The iterated transform acts by postprocessing the output of the DWT which makes the transform identification difficult. An iterated transform can also be used as a tool to make quantization changes in the transform coefficients and enable us to identify the transformation by the characteristics of its output. If such construction is numerically practical it provides an oblivious watermarking transform.

Combining iterated transforms could provide means of constructing watermarking transforms which affect specific regions of the image, perhaps the neighborhoods of the segmentation edges where local changes are least visible.

Any watermark, additive or multiplicative applied to A will depend on parameters controlling its strength, randomness and geometric properties. The range of those parameters is mostly established experimentally. The strength (regularization) parameter, for example, cannot be too small because the numerical detection of the resulting watermark will be difficult but also it cannot be too big because its visual detection in the image will be possible. It would be a viable optimization task to computationally establish the range of the strength parameter ε as a solution of an optimization problem which seeks the maximum value of ε which does not allow for visual detection of the watermark.

5 DWT-Projection Transform

As the simplest watermarking transformation based on the wavelet transform we consider the following iterated transform $\mathcal{I}$ in which the watermark $\mathcal{I}(A)$ depends globally on all DWT coefficients, however it modifies significantly only a small subset of wavelet coefficients.

Let $A = A^{(j)}$ be the matrix of approximation coefficients at level j resulting from applying the DWT $\mathcal{F}$ to a given image. We construct a random, key-dependent rank-one matrix d

and orthogonally decompose A into two orthogonal components as follows:

$$A = A_d + A_{d^\perp}$$

where

$$A_d = \frac{\langle A, d\rangle}{||d||^2} d \qquad A_{d^\perp} = A - A_d$$

We next define the projection enhancing transformation $\mathcal{I}$ as

$$\mathcal{I}(A) = (1+e)A_d + \delta A_{d^\perp}.$$

We select the parameters e, $0 \leq e \leq (||A|| - ||A_d||)/||A_d||$ and δ, $\delta = \delta(\epsilon)$ so as to preserve the norm of the coefficient matrix A:

$$||\mathcal{I}(A)|| = ||A||.$$

This implies that

$$\delta = \left(\frac{||A||^2 - (1+e)^2||A_d||^2}{||A||^2 - ||A_d||^2}\right)^{\frac{1}{2}}.$$

The projection enhancing transformation $\mathcal{I}$ amounts to increasing of a randomly (key-dependent) selected low-dimensional component A_d of A. The parameter e is the strength of the transformation (watermark). The magnitude of change in the transformed coefficients of A depends continuously on the size of e.

Finally in order to have a normalized strength parameter ε, $0 \leq \varepsilon \leq 1$ we set

$$\varepsilon = \frac{||A_d||}{||A|| - ||A_d||} e.$$

Consequently we obtain

$$\mathcal{I}(A) = \Big(1 - \delta - \varepsilon\big(1 - \frac{||A||}{||A_d||}\big)\Big)A_d + \delta A$$

where

$$\delta = \left(\frac{||A||^2 - ((1-\varepsilon)||A_d|| + \varepsilon||A||)^2}{||A||^2 - ||A_d||^2}\right)^{\frac{1}{2}}.$$

The experimentally recommended value of ε is $\varepsilon \approx 0.03$. This value is the maximal parameter strength yielding an invisible image watermark.

For the purpose of comparison we define a simplified projection enhancing transformation $\tilde{\mathcal{I}}$ which does not preserve the norm of A and only increases the component A_d.

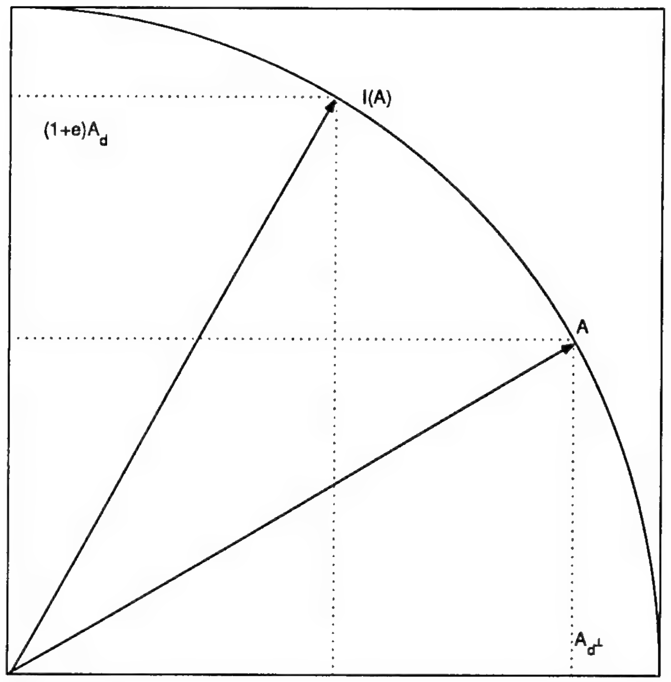


Figure 3: DWT-projection transformation

For $\varepsilon > 0$

$$\tilde{\mathcal{I}} = \varepsilon \frac{||A|| - ||A_d||}{||A_d||} A_d + A$$

where

$$A_d = \frac{\langle A, d \rangle}{||d||^2} d.$$

6 DWT-SVD Multiplicative Transforms

Let $A = A^{(j)}$ be the matrix of approximation coefficients at level j resulting from applying the DWT $\mathcal{F}$ to a given image. Our aim in constructing the DWT-SVD watermarking transform is to make the watermark $\mathcal{T}(A)$ depend globally on all transform coefficients. That is not the case when the watermark is a random additive perturbation of wavelet approximation coefficients in A.

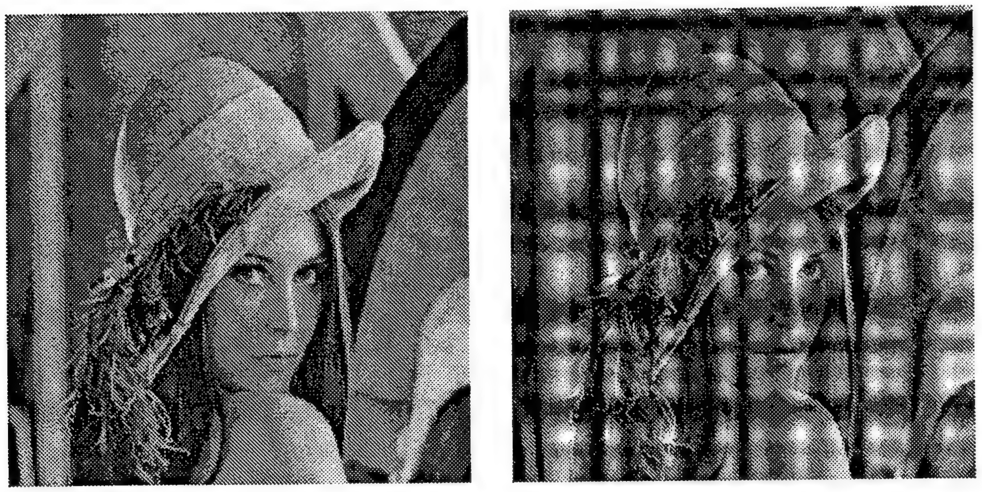


Figure 4: DWT-projection watermarks with $\varepsilon = 0.06$ and $\varepsilon = 0.8$

We compute the singular value decomposition

$$A = U\Sigma V^T.$$

The resulting orthogonal matrices U, V consisting of left and right singular vectors of A, and the diagonal matrix Σ of singular values depend globally and nonlinearly on all entries of A. A suitable modification of the SVD produces a watermark which depends globally and nonlinearly on A and which is image dependent. The geometric distortions of the original image (because of this complex structural dependence of the watermark on the image) have less potential of dislocating the watermark from its relative position in the image transform A and hence in the image itself. We begin by defining the DWT-SVD-SV transform $\mathcal{I}$. We define

$$\mathcal{I}(A) = \tilde{U}\,\Sigma\,\tilde{V}^T$$

where the orthogonal matrices $\tilde{U}$, $\tilde{V}$ are the modified matrices of singular vectors of A and are given by

$$\tilde{U} = O_U U \qquad \tilde{V} = O_V V$$

The matrices O_U and O_V are key dependent rotation matrices which are perturbation of the identity I. Clearly, the transform $\mathcal{I}$ preserves the norm of A. The strength of the perturbation in U and V which O_U and O_V induce is given by the parameter ε. Parameter value $\varepsilon > 0$ corresponds to orthogonal matrices O_U, O_V with determinant 1 such that $\varepsilon \to 0$ corresponds to $O_U, O_V \to I$. The dependence of O_U and O_V on ε is continuous. The recommended value of ε is $\varepsilon \approx 0.01$. This value is the maximal parameter strength yielding invisible image watermark.

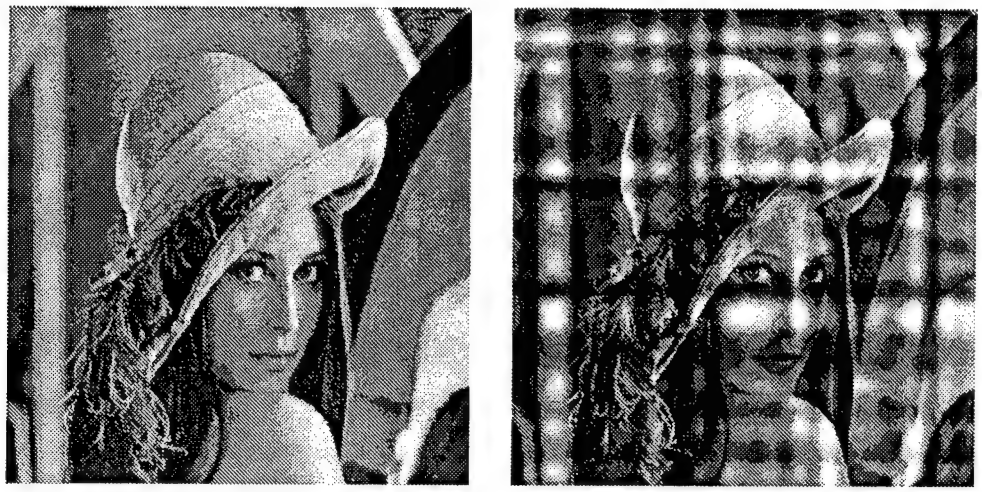


Figure 5: DWT-SVD watermarks with $\varepsilon = 0.01$ and $\varepsilon = 0.15$

We will next define the second DWT-SVD multiplicative transform $\tilde{\mathcal{I}}$ called DWT-SVD-D. This transform smoothly modifies the singular values by multiplying the diagonal matrix of singular values of the coefficient matrix by a key dependent orthogonal matrix R with determinant one. We set

$$\tilde{\mathcal{I}}(A) = U\tilde{\Sigma}V^T$$

where

$$\tilde{\Sigma} = R\Sigma.$$

Clearly, the transform $\tilde{\mathcal{I}}$ preserves the norm of A. As in the previous schemes the strength of the perturbation that the multiplication by $R = R(\varepsilon)$ induces is given by a parameter and $\varepsilon \to 0$, $R \to I$ when $\varepsilon > 0$. The recommended value of ε is $\varepsilon \approx 0.01$ which is the maximal parameter strength yielding an invisible image watermark.

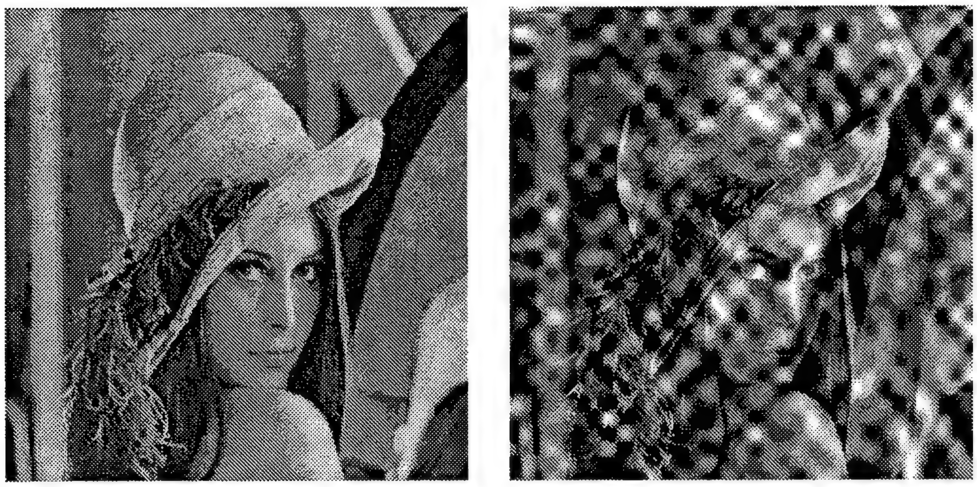


Figure 6: DWT-SVD-D watermarks with $\varepsilon = 0.015$ and $\varepsilon = 0.25$

7 Detection and Robustness Tests Results for Iterated Transforms Watermarks

In order to set the appropriate thresholds for the detection function in the iterated transform watermarking method, the original test image was marked with owner's watermark using the DWT-Projection transform (dwtpmark.m), DWT-SVD-SV transform (dwtsvdmark.m) and DWT-SVD-D transform (dwtsvddmark.m).

We used four basic detection methods for the purpose of thresholding and for robustness tests performed on our iterated transforms watermarks. The basic detection function was applied to: DWT coefficients of the image, the image, DCT (denoted as $\mathcal{D}$) of the DWT coefficients of the image, DCT of the image.

To abbreviate the above description of various detection procedures we set:

$C_{\mathcal{F}(M)} = C_{\mathcal{F}(M)}\left(\mathcal{I}(\mathcal{F}(M), k), \mathcal{F}(N)\right)$	detection in the DWT coefficients $\mathcal{F}(N)$ of the image N
$C_M = C_M\left(\mathcal{W}(M, k), N)\right)$	detection in the image N
$C_{\mathcal{D}(\mathcal{F}(M))} = C_{\mathcal{D}(\mathcal{F}(M))}\left(\mathcal{D}(\mathcal{I}(\mathcal{F}(M), k)), \mathcal{D}(\mathcal{F}(N))\right)$	detection in the DCT of the DWT coefficients of the image
$C_{\mathcal{D}(M)} = C_{\mathcal{D}(M)}\left(\mathcal{D}(\mathcal{W}(M, k)), \mathcal{D}(N)\right)$	detection in the DCT of the image

Table 1: Detection Thresholds for Iterated Transform Methods

Threshold	DWT-Projection	DWT-SVD-SV	DWT-SVD-D
$\tau_{\mathcal{I}}$	0.40	0.60	0.55
$\tau_{\mathcal{W}}$	0.55	0.65	0.65
$\tau_{\mathcal{DI}}$	0.55	0.70	0.50
$\tau_{\mathcal{DW}}$	0.60	0.60	0.50

In our thresholding experiments we set the owner's watermark to seed=100. For each of our watermarking transforms we plotted the values of the detection function

$$C_M(\mathcal{I}(\mathcal{F}(M),100),\mathcal{I}(\mathcal{F}(M),l)), \qquad 1 \leq l \leq 200$$

to set the detection threshold $\tau_{\mathcal{I}}$ for DWT coefficients.

We also plotted

$$C_M(\mathcal{W}(M,100),\mathcal{W}(M,l)), \qquad 1 \leq l \leq 200$$

to set the detection threshold $\tau_{\mathcal{W}}$ for watermarked images.

Consequently the experimental threshold for DWT coefficients satisfies

$$C_M(\mathcal{I}(\mathcal{F}(M),100),\mathcal{I}(\mathcal{F}(M),l)) \leq \tau_{\mathcal{I}} \quad 1 \leq l \leq 200, \quad l \neq 100$$

and the experimental threshold for images satisfies

$$C_M(\mathcal{W}(M,100),\mathcal{W}(M,l)) \leq \tau_{\mathcal{W}} \quad 1 \leq l \leq 200, \quad l \neq 100$$

The thresholding results for detection of the watermark in the DWT coefficient matrix and in the image reconstructed from watermarked DWT coefficients are given for the DWT-Projection transform in Figure 7 and in Figure 8, for the DWT-SVD-SV transform in Figure 9 and Figure 10 and for the DWT-SVD-D transform in Figure 11 and Figure 12.

All thresholding results are summarized in Table 1. In conclusion we make the following observations concerning the detectability of our iterated watermarks:

- iterated transform watermarks DWT-Projection, DWT-SVD and DWT-SVD-D are easily detectable using the standard detection function C_M
- the detection thresholds for image dependent iterated transforms watermarks are higher than the thresholds for the image independent random additive watermarks resulting in lower detectability of iterated transform watermarks

Our robustness testing followed exactly the methodology applied in [10] which was based on the work of J. Fridrich, see [7].

8 Conclusions

Two main classes of wavelet based watermarking methods dominate the image processing technology.

The methods in the first class generate random perturbations in the DWT coefficients and produce watermarks with controlled statistical characteristics.

The methods in the second class generate image dependent geometric transformations in the DWT coefficients and produce watermarks with controlled geometric characteristics. The iterated transforms methods introduced and studied in this project belong to the second group.

As a rule the detection thresholds for our image dependent methods are higher then for the wavelet-based methods from the first group which we studied in [10]. However all three watermarking methods studied here (including the simplest DWT-Projection method) have good detectability properties. The robustness properties, based on performed tests, are comparable to wavelet based methods studied in [10].

The most important results achieved in the development and implementation of iterated transform watermarking methods was obtaining:

- Crypto-wavelet type security provided by key dependent transform whose output differs significantly from the standard DWT but whose standard wavelet synthesis provides an invisible watermark.
- Watermarks depending nonlinearly and nonlocally on the image, preserving the gray level in the L^2 sense and providing a tool for the development of watermarking methods robust to specific geometric transformations (attacks).
- Watermarks with specific geometric characteristics (geometric quantization) corresponding to the strength parameter settings, providing a tool for the development of oblivious watermarking methods.
- Watermark robust with respect to wavelet compression.

Table 2: Robustness Testing Results for DWT-Projection Method

Test	**Detection Function**			
	$C_{\mathcal{F}(M)}$	C_M	$C_{\mathcal{D}(\mathcal{F}(M))}$	$C_{\mathcal{D}(M)}$
Blurring (number of iterations)	1	0	2	1
JPG Compression (quality units)	100 (no detection)	100 (no detection)	10	14

Table 3: Robustness Testing Results for DWT-SVD Method

Test	**Detection Function**			
	$C_{\mathcal{F}(M)}$	C_M	$C_{\mathcal{D}(\mathcal{F}(M))}$	$C_{\mathcal{D}(M)}$
Blurring (number of iterations)	2	0	1	2
JPG Compression (quality units)	9	89	100 (no detection)	9

Table 4: Robustness Testing Results for DWT-SVD-D Method

Test	**Detection Function**			
	$C_{\mathcal{F}(M)}$	C_M	$C_{\mathcal{D}(\mathcal{F}(M))}$	$C_{\mathcal{D}(M)}$
Blurring (number of iterations)	2	0	2	2
JPG Compression (quality units)	10	91	45	13

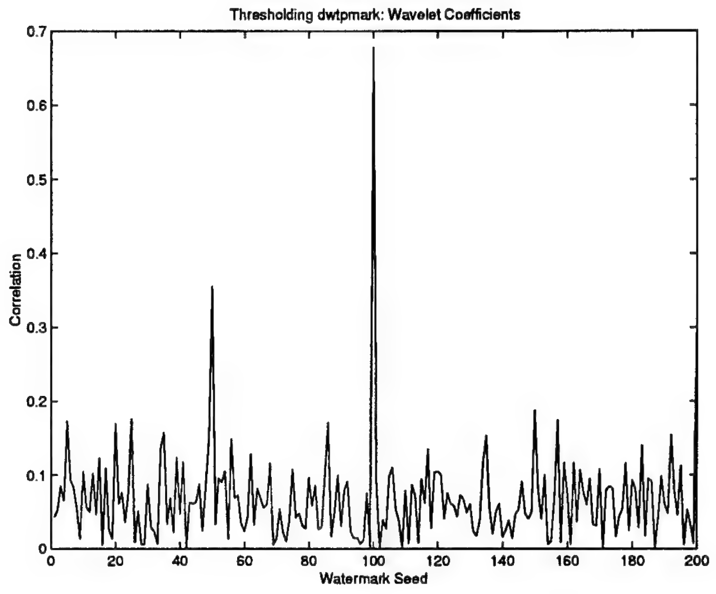


Figure 7: Detection of owner's watermark in the DWT coefficients (seed=100 among 200 random watermarks). DWT-Projection watermark (wavelet= db6, level=3, $\varepsilon = 0.03$).

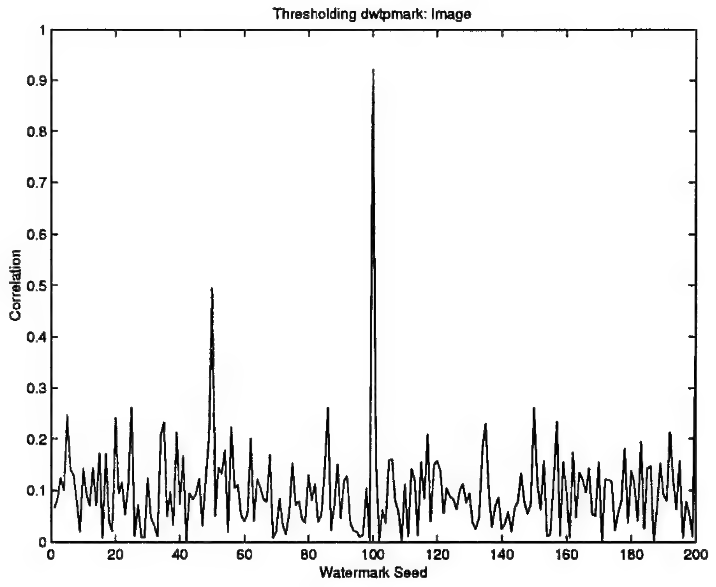


Figure 8: Detection of owner's watermark in the image (seed=100 among 200 random watermarks). DWT-Projection watermark (wavelet= db6, level=3, $\varepsilon = 0.03$).

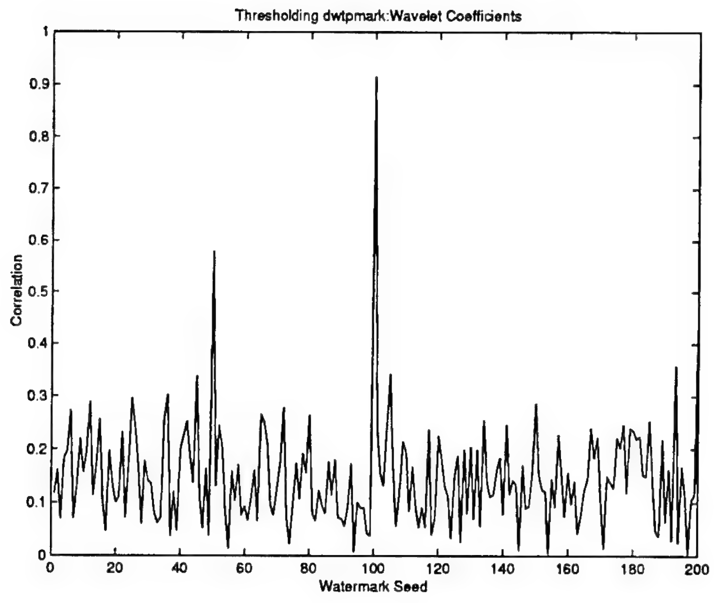


Figure 9: Detection of owner's watermark in the DWT coefficients (seed=100 among 200 random watermarks). DWT-SVD-SV watermark (wavelet= db6, level=3, $\varepsilon = 0.01$).

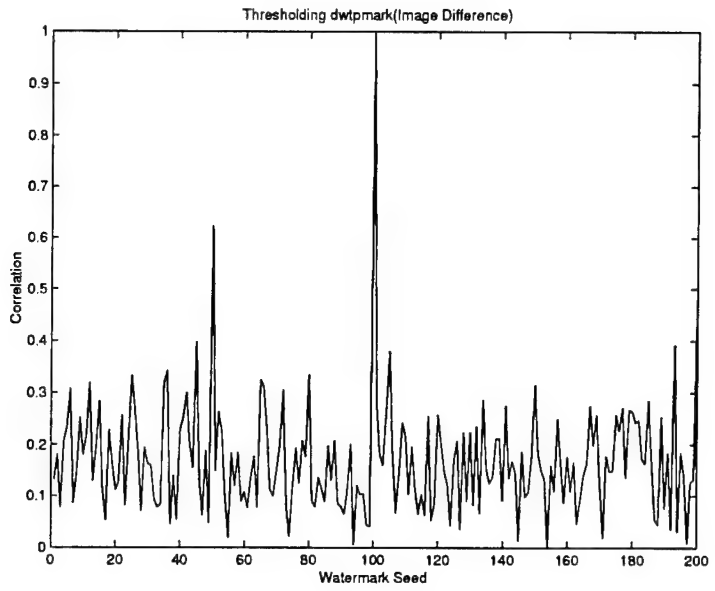


Figure 10: Detection of owner's watermark in the image (seed=100 among 200 random watermarks). DWT-SVD-SV watermark (wavelet= db6, level=3, $\varepsilon = 0.01$).

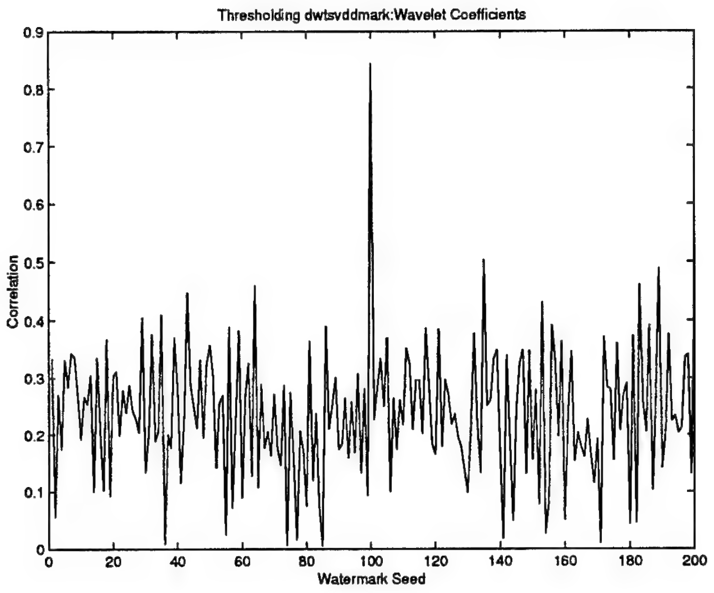


Figure 11: Detection of owner's watermark in the DWT coefficients (seed=100 among 200 random watermarks). DWT-SVD-D watermark (wavelet= db6, level=3, $\varepsilon = 0.01$).

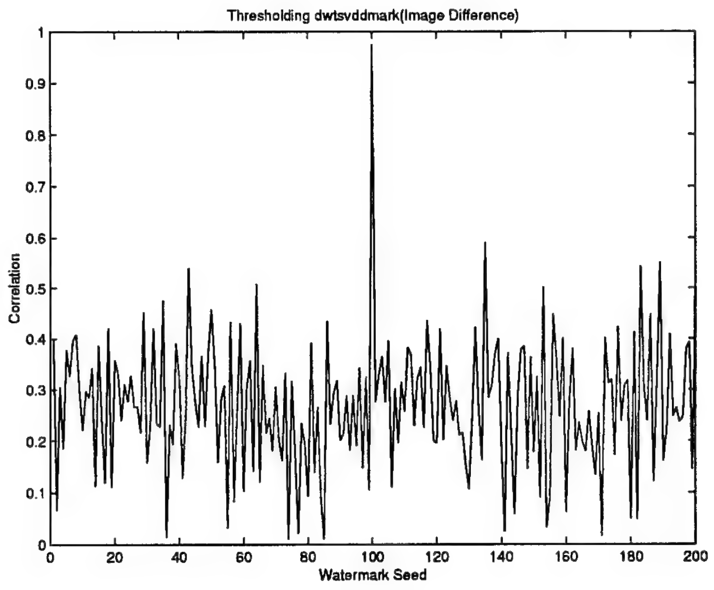


Figure 12: Detection of owner's watermark in the image (seed=100 among 200 random watermarks). DWT-SVD-D watermark (wavelet= db6, level=3, $\varepsilon = 0.01$).

References

[1] R. Dugad, Krishna Rataconda and Narendra Ahuja, *A New Wavelet Based Scheme for Watermarking Images* , University of Illinois at Urbana-Champain.

[2] Hisai Inoue, Akio Miyazaki, Akihiro Yamamoto and Takashi Katsura, *A Digital Watermark Based on the Wavelet Transform and its Robustness in Image Compression* , Matsushita Electric Industrial Co. Ltd, Kyushu University.

[3] J. Fridrich, *Robust Digital Watermarking Based on Key-Dependent Basis Functions*, The 2nd Information Hiding Workshop in Portland, Oregon, April 15-17, 1998.

[4] J. Fridrich, *Combining Low-frequency and Spread Spectrum Watermarking*, Proc. SPIE Int. Symposium on Optical Science, Engineering and Instrumentation, San Diego, July 19-24, 1998.

[5] J. Fridrich, *Image Watermarking for Tamper Detection*, Proc. ICIP'98, Chicago, October 1998.

[6] J. Fridrich, *Methods for Detecting Changes in Digital Images*, Proc. of the 6th IEEE International Workshop on Inteligent Signal Processing and Communication Systems (ISPACS'98), Melbourne, Australia, November 4-6, 1998.

[7] J. Fridrich, A. Baldoza and R. Simard, *On Digital Watermarks*, Rome Laboratory Report, 1997.

[8] D. Kundur and D. Hatzinakos, *A Robust Digital Image Watermarking Method Using Wavelet-Based Fusion*, to appear in Proc. Int. Conference in Image Processing, 1997.

[9] D. Kundur and D. Hatzinakos, *Digital Watermarking Based on Multiresolution Wavelet Data Fusion*, Proc. IEEE, Special Issue on Inteligent Signal Processing, under review, 1-43, 1997.

[10] A. Lutoborski, A. Baldoza and John Vergis, *On Wavelet-Based Method of Watermarking Digital Images*, Rome Laboratory Report, July 1998.

[11] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, San Diego, 1998.

[12] M. Misiti, Y. Misiti, G. Oppenheim and J.-M. Poggi, *Wavelet Toolbox*, The MathWorks Inc, Natick , Massachusetts, 1996.

[13] A. Piva, M. Barni, F. Bartolini and V. Cappelini, DCT-Based Watermark Recovering Without Resorting to the Uncorrupted Original Image, Proc. of the IEEE Int. Conference on Image Processing, Santa Barbara, California, October 26-29 1997, vol.1, p. 520-523.

[14] M.D. Swanson, M. Kobayashi and A.H. Tewfik, *Multimedia Data Embedding and Watermarking Technologies*, Invited Paper, to appear in the Proc. of the IEEE, 1998.

[15] M.D. Swanson, M. Kobayashi and A.H. Tewfik, *Robust Data Hiding for Images*, Proc. of the IEEE Digital Signal Processing Workshop, pp. 37-40, Loen, Norway, September 1996.

[16] M.D. Swanson, M. Kobayashi and A.H. Tewfik, *Transparent Robust Image Watermarking*, Proc. IEEE Int. Conf. on Image Processing, vol.3,pp. 211-214,1996.

[17] B. Tao and B. Dickinson, *Adaptive Watermarking in the DCT Domain*, Proc. IEEE Int. Conf. on Acoustics, Speach and Signal Processing, Munich, Germany, April 21-24, 1997.

[18] A.H. Tewfik, M.D. Swanson, B. Zhu, K. Hamdy and L. Boney, *Transparent Robust Watermarking for Images and Audio*, IEEE Trans. on Signal Proc., 1996.

[19] M.V. Wickerhauser, *Adapted Wavelet Analysis from Theory to Software*, AK Peters, Wellesley, Massachusetts, 1994.

[20] X.-G. Xia, C. Boncelet and G. Arce, *A Multiresolution Watermark for Digital Images*, Proc. of the ICIP, vol.1, pp.548-551, Santa Barbara, California, October 1997.

[21] Zixiang Xiong, Wengwu Zhu and Ya-Quin Zhang, *Multiresolution Watermarking for Images and Video: A Uniform Approach.*

9 Appendix. MATLAB Programs

dwtpmark

```
function [Y,cA_wat]=dwtpmark(input,seed,eps,wavelet,level)


% File:  dwtpmark.m
% Function dwtprmark applies a watermark to image 'input'
% and produces the watermarked image Y and a matrix of
% watermarked wavelet approximation coefficients cA_wat.
% It displays Figure(1)=input,Figure(2)=watermarked image
% Sample MATLAB command:  [Y,cA_wat]=dwtpmark('len256',0,0.03,'db6',3)
% the matrix of wavelet coefficients is projected onto a rank-one
% direction, projection modified with norm of wavelet coefficients preserved
% Parameters:
% input             image to be watermarked, bmp format assumed
% seed              PRNG seed to generate the projection direction
% eps               strength of the watermark, 0<eps<1,use eps<=0.03
% wavelet           wavelet class used in DWT
% level             level of DWT at which watermark is applied
% Created by: Dr. Adam Lutoborski, Syracuse University
% Date last modified:   November 11, 1999


% Read the input image, range [0,255],and its colormap
[X,map]=bmpread(input); figure(1);imshow(X,map);
% Perform wavelet decomposition
[C,S] = wavedec2(X,level,wavelet); cA=appcoef2(C,S,wavelet,level);
[M,N]=size(cA);


%set the seed for PRNG to seed
rand('seed',seed);


%orthogonal decomposition of approximation
%coefficients cA
```

```
d=rand(M,1)*rand(1,N); ncA=norm(cA,'fro'); nd=norm(d,'fro');
ncA_d=abs(trace(cA'*d))/nd; nwA=(1-eps)*ncA_d+eps*ncA;
cA_d=(trace(cA'*d)/(nd^2))*d;
delta=sqrt((ncA^2-nwA^2)/(ncA^2-ncA_d^2));
cA_wat=(1-delta-eps*(1-ncA/ncA_d))*cA_d+delta*cA;


% constructing watermarked image from
% watermarked coefficients cA_wat
cA_tilda=reshape(cA_wat,1,S(1,1)*S(1,2));
C(1,1:S(1,1)*S(1,2))=cA_tilda;
 Y=waverec2(C,S,wavelet);

bmpwrite(Y,map,'water.bmp');

figure(2);imshow(Y,map);
```

detectp

```
function [corr_coef,corr_im,corr_DCTcoef,corr_DCTim]=
detectp(input,original,seed,eps,wavelet,level)
% function  detects the presence of the wavelet based, key dependent
% watermark in the image 'input' using the original image 'original'
% and the secret key generated by seed.
%            Dr Adam Lutoborski, Syracuse University
% Date 7 September 1999

% reading in the data
[Z,map]=bmpread(input); [X,map]=bmpread(original);

% computing watermaked wavelet coefficients cA_wat for original image
[Y,cA_wat]=dwtpmark(original,seed,eps,wavelet,level);

% computing wavelet approximation coefficients cA_input
% for input image
[C,S]=wavedec2(Z,level,wavelet);
cA_input=appcoef2(C,S,wavelet,level);

% computing wavelet approximation coefficients cA
% for original image.
[C,S]=wavedec2(X,level,wavelet); cA=appcoef2(C,S,wavelet,level);

% comparing the coefficient watermark: cA_wat-cA with
% the coefficients of input-original: cA_input-cA
v=cA_wat-cA; watv=cA_input-cA;
corr_coef=abs(trace(v'*watv))/(norm(v,'fro')*norm(watv,'fro'));

disp(sprintf('corr_coef=%6.3f',corr_coef))
% comparing the dct of the coefficient watermark with
% the dct of the coefficients of input-original
DCTv=dct2(cA_wat-cA); DCTwatv=dct2(cA_input-cA);
DCTv=DCTv(1:min(32,max(size(DCTv))),1:min(32,max(size(DCTv))));
DCTwatv=DCTwatv(1:min(32,max(size(DCTwatv))),1:min(32,max(size(DCTwatv))));
DCTv(1,1)=0; DCTwatv(1,1)=0;
corr_DCTcoef=abs(trace(DCTv'*DCTwatv))/(norm(DCTv,'fro')*norm(DCTwatv,'fro'));


% comparing the image watermark Y-X with Z-X
m=Y-X; tm=Z-X;
corr_im=abs(trace(m'*tm))/(norm(m,'fro')*norm(tm,'fro'));
disp(sprintf('corr_im  =%6.3f',corr_im))



% comparing the dct of the image watermark with
% dct of input-original
DCTm=dct2(Y-X); DCTtm=dct2(Z-X); DCTm=DCTm(1:32,1:32);
```

```
DCTtm=DCTtm(1:32,1:32); DCTm(1,1)=0; DCTtm(1,1)=0;
corr_DCTim=abs(trace(DCTm'*DCTtm))/(norm(DCTm,'fro')*norm(DCTtm,'fro'));
```

PlotRandomCorrP

```
function plotRandomCorrP(original,eps, wavelet, level, seed,
MaxSeed);
%
% Plotting Correlations Between Random Watermarks
% Determines empirically the threshold to which to set the watermark detector.
% Uses the Wavelet-Based Watermark Detector developed by
% Dr. Adam Lutoborski, Syracuse University
% Arnold C. Baldoza, 2Lt, Air Force Research Laboratory/IFEC
% version 1.0
% 22 June 1998
%
% Parameters
% original          Image to be Watermarked and Detected on
% wavelet           Wavelet class to use in watermarking and detection
% level             Level of Wavelet decomposition
% seed              PRNG seed to generate the watermark patterns
% MaxSeed           Max PRNG seed to generate the random watermark patterns
%
corr_c=zeros(MaxSeed,1);     % correlation stored in vector corr
corr_i=zeros(MaxSeed,1);     % correlation stored in vector corr
corr_dc=zeros(MaxSeed,1);    % correlation stored in vector corr
corr_di=zeros(MaxSeed,1);    % correlation stored in vector corr

[Y,watCA2]=dwtpmark(original,seed,eps,wavelet,level);
[Z,map]=bmpread('water'); bmpwrite(Z,map,'test');

s = 1; for i=1:MaxSeed
   disp(['SEED = ' int2str(i) ]);
   [corr_coef,corr_im,corr_DCTcoef,corr_DCTim]=...
```

```
    detectp('test',original,s,eps,wavelet,level);

    corr_c(s) = corr_coef;
    corr_i(s) = corr_im;
    corr_dc(s) = corr_DCTcoef;
    corr_di(s) = corr_DCTim;
    s = s + 1;
end


figure(3); plot(abs(corr_c)); title ('Thresholding dwtpmark: Wavelet
Coefficients'); xlabel('Watermark Seed'); ylabel('Correlation');

figure(4); plot(abs(corr_i)); title ('Thresholding dwtpmark: Image
'); xlabel('Watermark Seed'); ylabel('Correlation');

figure(5); plot(abs(corr_dc)); title ('Thresholding
dwtpmark(DCT(Wavelet Coefficients))'); xlabel('Watermark Seed');
ylabel('Correlation');

figure(6); plot(abs(corr_di)); title ('Thresholding dwtpmark
(DCT(Image Difference))'); xlabel('Watermark Seed');
ylabel('Correlation');
```

dwtsvdmark

```
function [Y,cA_wat]=dwtsvdmark(input,seed,eps,wavelet,level)
% File:  dwtsvdmark.m
% Function dwtsvdmark applies a watermark to image 'input'
% and produces the watermarked image Y and a matrix of
% watermarked wavelet approximation coefficients cA_wat.
% It displays Figure(1)=input,Figure(2)=watermarked image
% Sample MATLAB command:  [Y,cA_wat]=dwtsvdmark('len256',0,0.01,'db6',3)
% Parameters:
% input        image to be watermarked, bmp format assumed
% seed         PRNG seed to generate the rotations of orthogonal matrices in SVD
% eps          strength of the watermark, 0<eps<1,use eps<=0.01
% wavelet      wavelet class used in DWT
% level        level of DWT at which watermark is applied
% Created by: Dr. Adam Lutoborski, Syracuse University
% Date last modified:   August 28, 1999



% Read the input image, range [0,255],and its colormap
[X,map]=imread(input,'bmp'); figure(1);imshow(X,map);
% Perform wavelet decomposition
[C,S] = wavedec2(X,level,wavelet); cA=appcoef2(C,S,wavelet,level);

[M,N]=size(cA); k=min([M,N]); rand('seed',seed);
%SVD and its watermaking modification
[U,Sigma,V]=svd(cA);

%watermarking modification of the SVD of cA

O_U=eye(M)+eps*(rand(M)-0.5); [q_U,r_U]=qr(O_U,0);
O_U=q_U*diag(sign(diag(r_U))); O_V=eye(N)+eps*(rand(N)-0.5);
[q_V,r_V]=qr(O_V,0); O_V=q_V*diag(sign(diag(q_V)));

cA_wat=O_U*U*Sigma*V'*O_V';

% constructing watermarked image from watermarked coefficients cA_tilda
cA_tilda=reshape(cA_wat,1,S(1,1)*S(1,2)); C(1,1:S(1,1)*S(1,2)) =
cA_tilda; Y=waverec2(C,S,wavelet);

%figure(5); imshow(Y,map);
imwrite(Y,map,'water.bmp','bmp'); figure(2);imshow(Y,map);
```

detectsvd

```
function [corr_coef,corr_im,corr_DCTcoef,corr_DCTim]=
detectsvd(input,original,seed,eps,wavelet,level)
% function  detects the presence of the wavelet based, key dependent
% watermark in the image 'input' using the original image 'original'
% and the secret key generated by seed.
%           Dr Adam Lutoborski, Syracuse University
% Date 7 September 1999

% reading in the data
[Z,map]=bmpread(input); [X,map]=bmpread(original);

% computing watermaked wavelet coefficients cA_wat for original image
[Y,cA_wat]=dwtsvdmark(original,seed,eps,wavelet,level);

% computing wavelet approximation coefficients cA_input
% for input image
[C,S]=wavedec2(Z,level,wavelet);
cA_input=appcoef2(C,S,wavelet,level);

% computing wavelet approximation coefficients cA
% for original image.
[C,S]=wavedec2(X,level,wavelet); cA=appcoef2(C,S,wavelet,level);

% comparing the coefficient watermark: cA_wat-cA with
% the coefficients of input-original: cA_input-cA
v=cA_wat-cA; watv=cA_input-cA;
corr_coef=abs(trace(v'*watv))/(norm(v,'fro')*norm(watv,'fro'));

disp(sprintf('corr_coef=%6.3f',corr_coef))
% comparing the dct of the coefficient watermark with
% the dct of the coefficients of input-original
```

```
DCTv=dct2(cA_wat-cA); DCTwatv=dct2(cA_input-cA);
DCTv=DCTv(1:min(32,max(size(DCTv))),1:min(32,max(size(DCTv))));
DCTwatv=DCTwatv(1:min(32,max(size(DCTwatv))),1:min(32,max(size(DCTwatv))));
DCTv(1,1)=0; DCTwatv(1,1)=0;
corr_DCTcoef=abs(trace(DCTv'*DCTwatv))/(norm(DCTv,'fro')*norm(DCTwatv,'fro'));


% comparing the image watermark Y-X with Z-X
m=Y-X; tm=Z-X;
corr_im=abs(trace(m'*tm))/(norm(m,'fro')*norm(tm,'fro'));
disp(sprintf('corr_im  =%6.3f',corr_im))



% comparing the dct of the image watermark with
% dct of input-original
DCTm=dct2(Y-X); DCTtm=dct2(Z-X); DCTm=DCTm(1:32,1:32);
DCTtm=DCTtm(1:32,1:32); DCTm(1,1)=0; DCTtm(1,1)=0;
corr_DCTim=abs(trace(DCTm'*DCTtm))/(norm(DCTm,'fro')*norm(DCTtm,'fro'));
```

PlotRandomCorrSVD

```
function plotRandomCorrSVD(original,eps, wavelet, level, seed,
MaxSeed);
%
% Plotting Correlations Between Random Watermarks
% Determines empirically the threshold to which to set the watermark detector.
% Uses the Wavelet-Based Watermark Detector developed by
% Dr. Adam Lutoborski, Syracuse University
% Arnold C. Baldoza, 2Lt, Air Force Research Laboratory/IFEC
% version 1.0
% 22 June 1998
%
% Parameters
% original          Image to be Watermarked and Detected on
% wavelet           Wavelet class to use in watermarking and detection
% level             Level of Wavelet decomposition
% seed              PRNG seed to generate the watermark patterns
% MaxSeed           Max PRNG seed to generate the random watermark patterns
%
corr_c=zeros(MaxSeed,1);    % correlation stored in vector corr
corr_i=zeros(MaxSeed,1);    % correlation stored in vector corr
corr_dc=zeros(MaxSeed,1);   % correlation stored in vector corr
corr_di=zeros(MaxSeed,1);   % correlation stored in vector corr

[Y,watCA2]=dwtsvdmark(original,seed,eps,wavelet,level);
[Z,map]=bmpread('water'); bmpwrite(Z,map,'test');


s = 1; for i=1:MaxSeed
   disp(['SEED = ' int2str(i) ]);
     [corr_coef,corr_im,corr_DCTcoef,corr_DCTim]=...
     detectsvd('test',original,s,eps,wavelet,level);

   corr_c(s) = corr_coef;
   corr_i(s) = corr_im;
   corr_dc(s) = corr_DCTcoef;
   corr_di(s) = corr_DCTim;
   s = s + 1;
end


figure(3); plot(abs(corr_c)); title ('Thresholding dwtpmark:Wavelet
Coefficients'); xlabel('Watermark Seed'); ylabel('Correlation');

figure(4); plot(abs(corr_i)); title ('Thresholding dwtpmark(Image
Difference)'); xlabel('Watermark Seed'); ylabel('Correlation');
```

```
figure(5); plot(abs(corr_dc)); title ('Thresholding
dwtpmark(DCT(Wavelet Coefficients))'); xlabel('Watermark Seed');
ylabel('Correlation');

figure(6); plot(abs(corr_di)); title ('Thresholding dwtpmark
(DCT(Image Difference))'); xlabel('Watermark Seed');
ylabel('Correlation');
```

dwtsvddmark

```
function [Y,cA_wat]=dwtsvddmark(input,seed,eps,wavelet,level)
% File:  dwtsvddmark.m
% Function dwtsvddmark applies a watermark to image 'input'
% and produces the watermarked image Y and a matrix of
% watermarked wavelet approximation coefficients cA_wat.
% It displays Figure(1)=input,Figure(2)=watermarked image
% Sample MATLAB command:  [Y,cA_wat]=dwtsvddmark('len256',0,0.01,'db6',3)
% the SVD of the matrix of wavelet coefficients is modified
% by multiplicative modification of singular values with norm preserved
% Parameters:
% input             image to be watermarked, bmp format assumed
% seed              PRNG seed to generate the projection direction
% eps               strength of the watermark, 0<eps,use eps<=0.01
% wavelet           wavelet class used in DWT
% level             level of DWT at which watermark is applied
% Created by: Dr. Adam Lutoborski, Syracuse University
% Date last modified:   November 11, 1999


% Read the input image, range [0,255],and its colormap
 [X,map]=imread(input,'bmp');
figure(1);imshow(X,map);
% Perform wavelet decomposition
```

```
[C,S] = wavedec2(X,level,wavelet); cA=appcoef2(C,S,wavelet,level);
[M,N]=size(cA); k=min([M,N]); cAmin=min(min(cA)); cAmax=max(max(cA));
rand('seed',seed);
%SVD and its watermaking modification
[U,Sigma,V]=svd(cA); sig=diag(Sigma);
%orthogonal perturbation of the diagonal in SVD and scaling by kap
R=eye(k)+eps*(rand(k)-0.5); [q_R,r_R]=qr(R,0);
R=q_R*diag(sign(diag(q_R))); sig_wat=R*sig;

cA_wat=U*diag(sig_wat)*V';

% constructing watermarked image from watermarked coefficients cA_tilda
cA_tilda=reshape(cA_wat,1,S(1,1)*S(1,2)); C(1,1:S(1,1)*S(1,2)) =
cA_tilda; Y=waverec2(C,S,wavelet); imwrite(Y,map,'water.bmp','bmp');
figure(2);imshow(Y,map);
```

detectsvdd

```
function [corr_coef,corr_im,corr_DCTcoef,corr_DCTim]=
detectsvdd(input,original,seed,eps,wavelet,level)
% function  detects the presence of the wavelet based, key dependent
% watermark in the image 'input' using the original image 'original'
% and the secret key generated by seed.
%           Dr Adam Lutoborski, Syracuse University
%           Date 10 October 1999

% reading in the data
[Z,map]=bmpread(input); [X,map]=bmpread(original);

% computing watermaked wavelet coefficients cA_wat for original image
[Y,cA_wat]=dwtsvddmark(original,seed,eps,wavelet,level);

% computing wavelet approximation coefficients cA_input
% for input image
```

```
[C,S]=wavedec2(Z,level,wavelet);
cA_input=appcoef2(C,S,wavelet,level);

% computing wavelet approximation coefficients cA
% for original image.
[C,S]=wavedec2(X,level,wavelet); cA=appcoef2(C,S,wavelet,level);

% comparing the coefficient watermark: cA_wat-cA with
% the coefficients of input-original: cA_input-cA
v=cA_wat-cA; watv=cA_input-cA;
corr_coef=abs(trace(v'*watv))/(norm(v,'fro')*norm(watv,'fro'));
disp(sprintf('corr_coef=%6.3f',corr_coef))
% comparing the dct of the coefficient watermark with
% the dct of the coefficients of input-original
DCTv=dct2(cA_wat-cA); DCTwatv=dct2(cA_input-cA);
DCTv=DCTv(1:min(32,max(size(DCTv))),1:min(32,max(size(DCTv))));
DCTwatv=DCTwatv(1:min(32,max(size(DCTwatv))),1:min(32,max(size(DCTwatv))));
DCTv(1,1)=0; DCTwatv(1,1)=0;
corr_DCTcoef=abs(trace(DCTv'*DCTwatv))/(norm(DCTv,'fro')*norm(DCTwatv,'fro'));


% comparing the image watermark Y-X with Z-X
m=Y-X; tm=Z-X;
corr_im=abs(trace(m'*tm))/(norm(m,'fro')*norm(tm,'fro'));
disp(sprintf('corr_im  =%6.3f',corr_im))



% comparing the dct of the image watermark with
% dct of input-original
DCTm=dct2(Y-X); DCTtm=dct2(Z-X); DCTm=DCTm(1:32,1:32);
DCTtm=DCTtm(1:32,1:32); DCTm(1,1)=0; DCTtm(1,1)=0;
corr_DCTim=abs(trace(DCTm'*DCTtm))/(norm(DCTm,'fro')*norm(DCTtm,'fro'));
```

PlotRandomCorrSVDd

```
function plotRandomCorrSVDd(original,eps, wavelet, level, seed,
MaxSeed);
%
% Plotting Correlations Between Random Watermarks
% Determines empirically the threshold to which to set the watermark detector.
% Uses the Wavelet-Based Watermark Detector developed by
% Dr. Adam Lutoborski, Syracuse University
% Arnold C. Baldoza, 2Lt, Air Force Research Laboratory/IFEC
% version 1.0
% 11 November 1999
%
% Parameters
% original          Image to be Watermarked and Detected on
% wavelet           Wavelet class to use in watermarking and detection
% level             Level of Wavelet decomposition
% seed              PRNG seed to generate the watermark patterns
% MaxSeed           Max PRNG seed to generate the random watermark patterns
%
corr_c=zeros(MaxSeed,1);    % correlation stored in vector corr
corr_i=zeros(MaxSeed,1);    % correlation stored in vector corr
corr_dc=zeros(MaxSeed,1);   % correlation stored in vector corr
corr_di=zeros(MaxSeed,1);   % correlation stored in vector corr

[Y,watCA2]=dwtsvddmark(original,seed,eps,wavelet,level);
[Z,map]=bmpread('water'); bmpwrite(Z,map,'test');


s = 1; for i=1:MaxSeed
   disp(['SEED = ' int2str(i) ]);

   [corr_coef,corr_im,corr_DCTcoef,corr_DCTim]=...
```

```
    detectsvdd('test',original,s,eps,wavelet,level);

    corr_c(s) = corr_coef;
    corr_i(s) = corr_im;
    corr_dc(s) = corr_DCTcoef;
    corr_di(s) = corr_DCTim;
    s = s + 1;
end


figure(3); plot(abs(corr_c)); title ('Thresholding
dwtsvddmark:Wavelet Coefficients'); xlabel('Watermark Seed');
ylabel('Correlation');

figure(4); plot(abs(corr_i)); title ('Thresholding dwtsvddmark(Image
Difference)'); xlabel('Watermark Seed'); ylabel('Correlation');

figure(5); plot(abs(corr_dc)); title ('Thresholding
dwtsvddmark(DCT(Wavelet Coefficients))'); xlabel('Watermark Seed');
ylabel('Correlation');

figure(6); plot(abs(corr_di)); title ('Thresholding dwtsvddmark
(DCT(Image Difference))'); xlabel('Watermark Seed');
ylabel('Correlation');
```

IMPLEMENTATION OF PETRI-NETS BASED MULTI-SOURCE ATTACK DETECTION MODEL

Brajendra Panda
Associate Professor
Department of Computer Science

University of North Dakota
P.O. Box 9015
Grand Forks, ND 58202-9015

Final Report
for
Summer Research Extension Program
Rome Research Site

Sponsored by:

Air Force Office of Scientific Research
Bolling Air Force Base
Washington, D.C.

and

Rome Research Site
Air Force Research Laboratory
Rome, NY 13440

February 2000

Implementation of Petri-Nets Based Multi-Source Attack Detection Model

Brajendra Panda
Associate Professor
Department of Computer Science
University of North Dakota

Abstract

The Information Assurance (IA) program was started by the US Air Force (USAF) with a view to protect USAF's information systems, to detect any information attacks on these resources, and to react to situations that can interrupt operations of such systems. Intrusion detection plays a very critical role in this process. Although existing intrusion detection systems can easily detect copycat types of attacks, they fail to identify prudent coordinated attacks originating from multiple sources. Therefore, development of a system that can quickly and accurately recognize multi-source attacks is extremely essential for survival of information systems. During my research at the Rome Research Site in summer of 1998, I had proposed a conceptual model to analyze sensor data for detection of coordinated attacks on computer systems [Pand98]. These attacks originate from multiple sites and therefore named as multi-source attacks. Petri-nets were used to detect such attacks. We have implemented the model using C and Visual C++. In this paper, we provide the implementation particulars of the project.

Implementation of Petri-Nets Based Multi-Source Attack Detection Model

Brajendra Panda

1. Introduction

The last decade of the last millennium has witnessed incredible revolution in computer technology. The most popular event has been the evolution of the Internet, which has become the indispensable media for almost every computer user for collection and dissemination of information. However, this remarkable tool also provides malicious users the opportunity to illegally access others computer systems. The objective of an attacker could range from causing momentary disorder in the organizational activities to complete destruction of the opponent's information system. Defensive Information Warfare (DIW) plays a vital role in the protection of computer systems from unauthorized access. The US Air Force's Information Assurance (IA) program is heavily involved in research and development of dependable systems for its command, control, communications, and computer (C^4) systems [Metc97].

A lot of research, including strict access control mechanisms, has been performed to protect systems from unauthorized users. Some of the related research done by the investigator can be found in [Pand94], [Pan95a], and [Pan95b]. Ammann et al. [Amm97] and Graubart et al [Grau96] have discussed some of these issues with regard to information warfare environment. However, these techniques are not sufficient to provide absolute protection from information attacks. When protection mechanisms for these systems fail, immediate detection of an intrusion, and fast and accurate recovery of affected systems become highly important [Pand99].

A survey of existing intrusion detection mechanisms is presented in [Lunt93]. In [Goan99] a method is presented on collection and evaluation of intrusion evidence. Durst et.al. have presented an approach to evaluate intrusion detection systems in [Durs99]. Although existing intrusion detection systems can easily detect copycat types of attacks, they fail to identify prudent coordinated attacks originating from multiple sources. Therefore, development of a system that can quickly and accurately recognize multi-source attacks is extremely essential for survival of information systems. In this paper, we discuss implementation of a model developed in [Pand98] for analysis of sensor data to determine attacks that are launched from several distributed sites.

The rest of the report is organized as follows. In section 2, we specify the motivation for this research. A brief discussion of the model is presented in section 3. Section 4 offers implementation facts of the model. Section 5 discusses results and conclusion of the project. The appendix section of this paper provides the source code.

2. Motivation

A defensive information warfare decision support system called Extensible Prototype for Information Command and control (EPIC) is currently under development at Rome Research Site. This system is being designed to monitor system and network parameters to identify if an attack occurs and when it occurs to suggest necessary actions to be taken. Data obtained from various sensors at this site are passed through an expert system called G2, which then stores essential data in a database. Examples of information collected and stored in the database include user ids and passwords, login time, logout time, remote system id (from where access was made), id of the system that was accessed, commands executed, etc. If any suspicious activity by a user is found, the expert system will alert the security administrators of the event. Such activities are usually determined by examining various commands executed by users. Using this system it would be easy to detect copycat types of attacks launched by a single attacker. However, any coordinated attacks organized by a group of sophisticated attackers will be impossible to detect without an efficient, optimized, and refined process that requires automated correlation of data obtained from various sensors. This issue motivated us to develop a conceptual model based on Petri-nets that monitors commands of all users in the system and correlates them in order to find any attack patterns. The model is briefly described in the next section. Interested readers may refer to [Pand98] for further details.

3. The Model

As per [Pand98], Figure 1 below depicts an attack pattern, which requires commands C_1 and C_2, executed in any order, and are followed by command C_3. Then commands C_5, C_6 and C_7 need to be executed without any particular ordering among them. However, command C_4 must be executed before execution of command C_7. Finally, execution of command C_8 would complete the attack. Using the model developed in [Pand98], this attack pattern can then be transformed to a Petri-Net as shown in Figure 2. A background on Petri-Nets and the process of developing the structure is discussed in [Pand98]. A detailed discussion on Petri-Nets can be found in [Mura89].

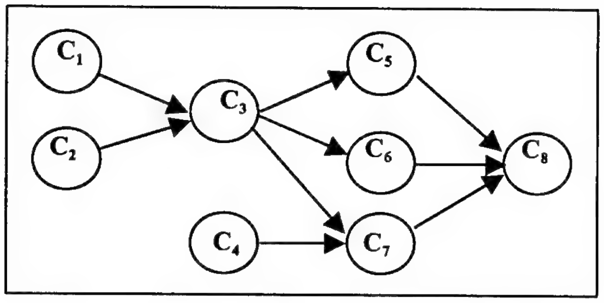


Figure 1: A Graph Representing Attack

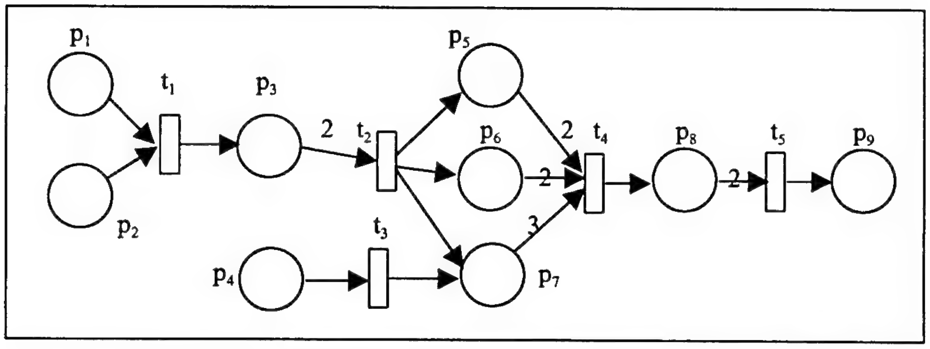


Figure 2: A Petri-Net Representing the Attack Shown in Figure 1.

4. Implementation Facts

In this project, the Petri-Net attack detection model described in previous section is implemented using 'C' language. Visual C++ is used to offer a graphical interface. Following is the brief description of the files used:

Attack Pattern Files: The attack pattern is represented in a file format using 'p's (place) and 't's (transition). For example t_2, p_5, p_6, p_7, t_4 indicates that the places p_5, p_6, p_7 can not be enabled (setting to 1) until and unless t_2 is fired. Similarly t_4 can not be fired, if any one of the places is not yet enabled. In general, before firing a transition all its immediate predecessor places has to be enabled. Note that all the immediate predecessor places need not be in the same line in a file. Similarly before enabling a place, its immediate predecessor transactions has to be fired. The list of immediate predecessors for a place or transition can be obtained from these attack pattern files. If there is more than one place in a line, they are stored in increasing value of the place number. Each place in the Petri-Net model corresponds to the command number. These files have .dat extension.

Input Data File: The sequences of command numbers are stored in this (inp_data) file.

Command User Information File: The user id is stored against the command number (issued by the same user) in this (com_user) file.

Output Files: Compl_patts contains the list of completely attacked patterns detected and similarly devel_patts contains developing patterns list. Sequence of commands forming a partial attack pattern (at least half but not full) is treated as developing pattern). Finally the commands and users ids associated with these attack patterns are stored in files of the format patt_. For example, pattern1 information is stored in patt1.

There functions used to detect distributed attacks are described briefly below:

file_max_p_t(): This function takes the file number as a parameter and returns the maximum place number and transition number found in that file. Typically each file contains an attack pattern. The given file is scanned one line at a time until the last line is reached, which contains the maximum place number and maximum transition number. Note that if there is more than one place in a line they are stored in increasing value of the place number and it is obvious that the transition with the maximum value is found in the last line.

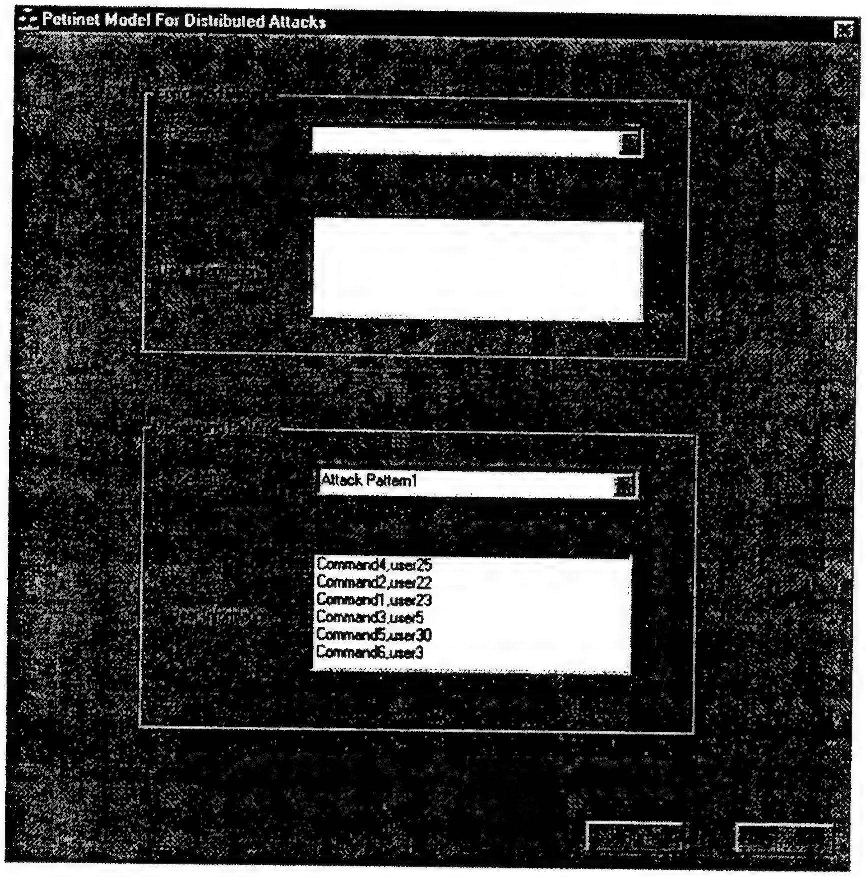
enable_place(): This function takes the place number and file number as parameters and determines whether that place can be enabled or not. The function determines the list of transactions that should be fired before this place can be enabled for a particular attack pattern. (Attack pattern stored in the file is used for detection purpose). If any of the transitions in the list is not yet fired the function returns 0 indicating the place under consideration can not be enabled and returns 1 if all the transitions in the list are fired.

check_trans_list(): When a place is enabled, it has to be checked whether enabling of this place triggers firing of any transition(s). check_trans_list() function returns the list of transitions that may be fired on accounting of enabling a place. This function obviously takes place number and file number (for attack pattern) as parameters and returns the list of transitions in a global integer array.

enable_trans(): This function takes the transition number and file number as parameters and returns 0 if the transition can not be fired. The function obtains the list of places that have to be enabled for firing this particular transition. If all the places are enabled it returns the transition number and if any of the places are not yet enabled it returns 0.

print_user_info(): This function reads the command number and determines the user id from user_info file and writes the output (command number and user id) into a file determined by the second parameter.

main(): The main function opens all the input and output files. It initializes the various variables and integer arrays. Then it reads the command numbers sequentially from the file. For each command read it checks whether the place can be enabled for a particular attack pattern using enable_place() function. If the place can be enabled, it calls check_trans_list(), to determine the list of transitions that has to checked for triggering. For each transition in the list, enable_trans() function is called to see if a particular transition can be triggered. The whole process mentioned above is repeated for each attack pattern. Finally output is written to appropriate files. The Visual C++ program reads from these files and displays the results in graphical form.



```
Command4,user25
Command2,user22
Command1,user23
Command3,user5
Command5,user30
Command6,user3
```

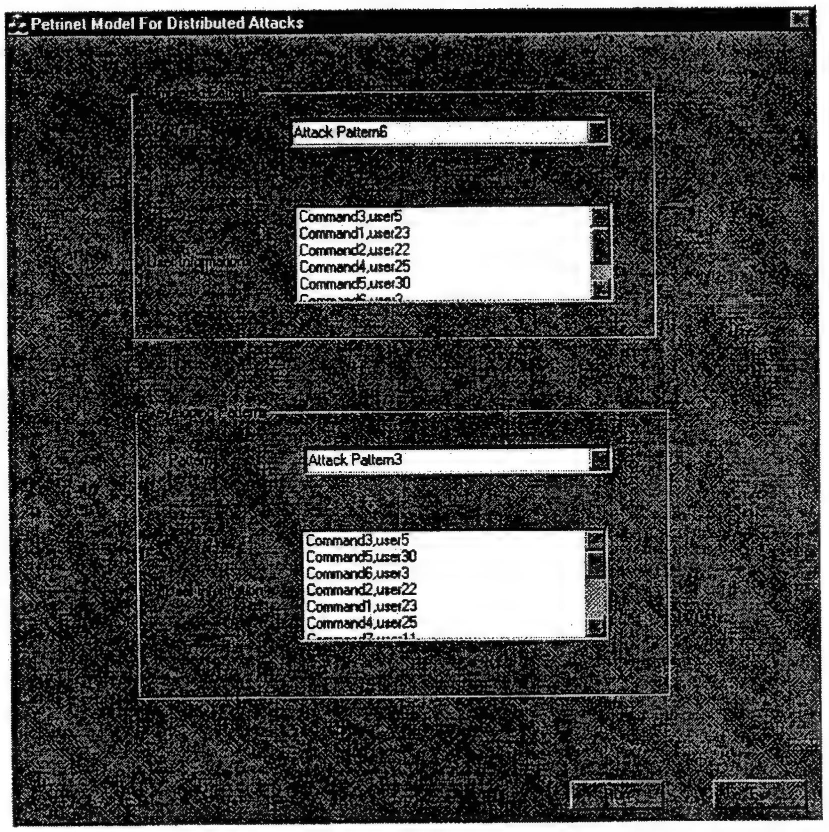
Figure 3: Output Showing Developing Attack Patterns

5. Results and Conclusion

During the execution of the program, many sets of command sequences were passed as input and results were verified. It must be noted that the program was run using imaginary attack patterns. In all cases of the program execution, the model was able to detect both completed attack patterns as well as developing patterns. A pattern was considered developing when at least 50% of its command sequences starting from the beginning of the

Petri-Net were executed. Results of two different executions are included in figures 3 and 4. Figure 3 shows an output where the sequence of commands form four developing patterns and no completed patterns. Another output with four completed attack patterns and two developing patterns are illustrated in Figure 4.

Our program used six imaginary attack patterns. Each output was in full conformity with expected results. The source code of the program is provided in the appendix. With minor modification, the program can be used to detect any number of attack patterns.



Attack Pattern6

```
Command3,user5
Command1,user23
Command2,user22
Command4,user25
Command5,user30
```

Attack Pattern3

```
Command3,user5
Command5,user30
Command6,user3
Command2,user22
Command1,user23
Command4,user25
```

Figure 4: Output Showing Both Developing and Completed Attack Patterns

Acknowledgments

I acknowledge and thank Mr. Joseph Giordano of Rome Research Site, Air Force Research Laboratory for his support of during this research. I am also thankful to Mr. Rajesh Yalamanchili for working tirelessly to finish up implementation of this project.

References

[Amm97] P. Ammann, S. Jajodia, C. D. McCollum, and B. Blaustein, "Surviving Information Warfare Attacks on Databases", In Proceedings of the 1997 IEEE Symposium on Security and Privacy, p.164-174, Oakland, CA, May 1997.

[Durs99] R. Durst, et.al, "Testing and Evaluating Computer Intrusion Detection Systems", In Communications of the ACM, Vol. 42, No. 7, p. 53-61,July 1999.

[Goan99] T. Goan, "A Cop on the Beat: Collecting and Appraising Intrusion Evidence", In Communications of the ACM, Vol. 42, No. 7, p. 46-52, July 1999.

[Grau96] R. Graubart, L. Schlipper, and C. McCollum, "Defending Database Management Systems Against Information Warfare Attacks", Technical report, The MITRE Corporation, 1996.

[Lunt93] T. F. Lunt, "A Survey of Intrusion Detection Techniques", Computers & Security, Vol. 12, No. 4, p. 405-418, June 1993.

[Metc97] Therese R. Metcalf, "Computer Security Assistance Program For The Twenty-First Century (CSAP21) Functional Requirements – Draft", Technical Report, The MITRE Corporation, Bedford, MA, December 1997.

[Mura89] Tadao Murata, "Petri Nets: Properties, Analysis and Applications", In Proceedings of the IEEE, Vol. 77, No. 4, p. 541-580, April 1989.

[Pand94] B. Panda, W. Perrizo, and R. Haraty, "Secure Transaction Management and Query Processing in MLS Database Systems", In Proceedings of 1994 ACM Symposium on Applied Computing, pp. 363-368, Phoenix, AZ.

[Pan95a] B. Panda and W. Perrizo, "Query Execution in PRISM and SeaView: A Cost Analysis", In Proceedings of 1995 ACM Symposium on Applied Computing, San Jose, CA.

[Pan95b] B. Panda and W. Perrizo, "Maintaining Surrogate Data for Query Acceleration in Multilevel Secure Database Systems", In Lecture Notes in Computer Science, No. 1006, Information Systems and Data Management, Editor: Subhash Bhalla, Springer-Verlag, November 1995.

[Pand98] B. Panda, "A Model to Analyze Sensor Data For Detection of Multi-Source Attacks", Final Report, AFOSR Summer Faculty Research Program, AFRL, Rome Research Site, Rome, August 1998.

[Pand99] B. Panda and Joseph Giordano, "Defensive Information Warfare", Guest Editors' Article, In Communications of the ACM, Vol. 42, No. 7, p. 31-32, July 1999.

Appendix

```
/* This program reads a set of numbers (that corresponds to command numbers) sequentially and checks to see if
they are producing an attack  patterns. Attack patterns stored in files is read by the program and used for detection
purposes. Whenever half the number of commands to complete an attack pattern are read (executed), it is
considered as developing pattern and is displayed. When all the commands needed are read (executed) it is
displayed as Completed attack pattern. */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define max_len 40            // Maximum length of any line in input files
                              // storing attack patterns
#define att_patterns 6     // Number of attack patterns
#define max_places 20      // Max. number of places in any attack pattern
#define max_transitions 20 // Max. number of transitions in any attack pattern
```

```
int file_max_p_t(int file_num );
int enable_place (int a, int fil_num);
void check_trans_list (int b, int fi_num);
int enable_trans (int c, int f_num);
void print_user_info(int d, int e);

FILE *fptr[att_patterns],*patt_fptr[att_patterns],*com_user_fptr;
int t[att_patterns][max_transitions], p[att_patterns][max_places];
int max_p_val[att_patterns],max_t_val[att_patterns];
int transtn_list[max_transitions];

int main()
{
 FILE *data_fptr, *compl_fptr, *devel_fptr;
 int i,j,m,com_num,trans_num,flag1,flag2=0,compl_flag[att_patterns];
 int com_seq[att_patterns][max_places],com_seq_num[att_patterns];
 int seq_end_flag[att_patterns],developing_flag[att_patterns];
 float x;
 char out_file[20];

/* The following for loop calls the file_max_p_t function repeatedly,
   which inturn determines the maximum place number and maximum transition
   number for a particular attack pattern, stored in file format */

 for(i=0;i<att_patterns;i++)
   file_max_p_t(i);

 if ((compl_fptr=fopen("compl_patts","w"))==NULL)
 {
   printf("file compl_patts not found!!\n");
   return 0;
 }
 if ((devel_fptr=fopen("devel_patts","w"))==NULL)
 {
   printf("file devel_patts not found!!\n");
   return 0;
 }
 if ((com_user_fptr=fopen("com_user","r"))==NULL)
 {
   printf("file com_user not found!!\n");
   return 0;
 }

 for(i=0;i<att_patterns;i++)
 {
   sprintf(out_file,"%s%d","patt",(i+1));
   if ((patt_fptr[i]=fopen(out_file,"w"))==NULL)
   {
     printf("file %s not found!!\n",out_file);
     return 0;
   }
 }
/* The following for loop initializes the place number values to Zero */
```

```
 for(m=0;m<att_patterns;m++)
 {
   for(i=0;i<max_places;i++)
   {
        p[m][i]=0;
   }
 }

/* The following for loop initializes the transition number values to zero */

 for(m=0;m<att_patterns;m++)
 {
   for(i=0;i<max_transitions;i++)
   {
        t[m][i]=0;
   }
 }

 for(i=0;i<att_patterns;i++)
 {
   t[i][0]=1;          // First column of transition  number matrix is
                              // set to '1'.
   seq_end_flag[i]=0;  // seq_end_flag which indiactes whether a
                              // particular attack pattern is completed is
                              // initialized to zero.
   com_seq_num[i]=0;   // com_seq_num which gives a count of commands
                              // which are read (executed) and match with a
                              // particular attack pattern
   developing_flag[i]=1; // developing_flag which indiactes whether a
                              // particular attack pattern is half way
                              // through is initialized to one.
   compl_flag[i]=0;    // compl_flag which indiactes whether a
                              // pattern is completed is initialized to '0'
 }

/* The file containing sets of numbers (corresponding  to commands) is
   being opened and read sequentially to see if it is producing any attack
   patterns */

 if ((data_fptr=fopen("inp_data","r"))==NULL)
    printf("file inp_data not found!!\n");
 else
 {
  fscanf(data_fptr,"%d",&com_num);
  while (!feof(data_fptr))
  {

/* The following for loop checks whether the command number read from
   input file (inp_data) contributes to the formation of any of the attack
   patterns */

     for(m=0;m<att_patterns;m++)
     {
```

```
/* The following if condition checks whether the command number read is
   less than the maximum value possible in a particular attack pattern,
   and the place number of the corresponding number is not yet set and
   finally checking if the place can be enabled */
          if((com_num<=max_p_val[m])&&(p[m][com_num]==0)&&(enable_place(com_num,m)==1))
          {
            p[m][com_num]=1; // setting the place to '1'
            com_seq[m][com_seq_num[m]]=com_num; // noting down the
                                                  // developing sequence
        com_seq_num[m]++;
            for(i=0;i<max_transitions;i++)
                   transtn_list[i]='\0';

/* The following function returns the list of transactions (in an array,
   transtn_list[]) that has to be checked to see if they can be enabled */

            check_trans_list(com_num,m);

/* The enable_trans() function returns non zero if any transition can be
   enabled. This function is called for each element in transtn_list
   array */

        for(j=0;transtn_list[j]!='\0';j++)
        {
         trans_num=enable_trans(transtn_list[j],m);
            if(trans_num>0)
            {
                   t[m][trans_num]=1;  // setting the transition to '1'

/* The following if checks whether an attack pattern is completed and if
   it is completed, prints that particular pattern as completed attack
   pattern */

           if((trans_num==max_t_val[m])&&(compl_flag[m]==0))
                   {
                     compl_flag[m]=1;
                     developing_flag[m]=0; // It's no more a developing pattern
             fprintf(compl_fptr,"%s%d\n","Attack Pattern",m+1);

/* The following for loop prints the completed attack pattern */

                     for(i=0;i<(max_p_val[m]);i++)
                     {

                            print_user_info(com_seq[m][i], m); // prints the user info

                     }
                   } // end of if that checks for completed pattern
         } // end of if which enables a transition
       }// end of for which checks all the possible transitions that can be    // enabled
           } // end of if which enables a place
    } // end of for
   fscanf(data_fptr,"%d",&com_num);
```

```
    } // end of while which reads the place numbers

 /* printing developing patterns */

    flag1=0;
    for(m=0;m<att_patterns;m++)
    {

 /* Any pattern which is half through is treated as developing pattern. The
   following if loop checks whether it is half way through but not yet
   completed and then prints it as developing pattern */

          x= (float) max_p_val[m]/2;
               if((com_seq_num[m] >= x)&&(developing_flag[m]==1))
               {
             fprintf(devel_fptr,"%s%d\n","Attack Pattern",m+1);

                      for(i=0;i<com_seq_num[m];i++)
                      {
                         print_user_info(com_seq[m][i], m); // prints user info
                      }

               } // end of if
    } // end of for
 /* end of printing developing patterns */

    fclose(data_fptr);
  } // end of else which indicates that data file is found.
    for(i=0;i<att_patterns;i++)
    {
       fclose(fptr[i]);
       fclose(patt_fptr[i]);
    }
    fclose(compl_fptr);
    fclose(devel_fptr);
    fclose(com_user_fptr);
  return 0;
 } // end of main

 /*  The following function takes the place number and file number (for
    attack pattern, please see the note following) as input and determines
    whether that place can be enabled or not.
    Note: Each attack pattern is stored in a file. For example, attack
    pattern 1 is stored in file pt1.dat. The function retrives the
    corresponding attack pattern from the file to check if the place
    number passed can be enabled. */

 int enable_place (int a, int fil_num)
 {
  int j,flag1;
  int set_place,trans_ctr,trans_list[max_transitions],start_trans_num;
  char line_p[max_len],place_name[10], *start_trans;

  rewind(fptr[fil_num]);
```

```
/* The trans_list array stores the list of transitions that are
   immediately before the place number being checked. The following for
   loop declares and initializes the array. */

  for(j=0;j<max_t_val[fil_num];j++)
       trans_list[j]='\0';
  sprintf(place_name,"%c%d%c",'p',a,',');
  fscanf(fptr[fil_num],"%s",line_p); // opening the required attack pattern
  trans_ctr=0; // initializing the number of transitions to '0'
  flag1=0;

  while(!feof(fptr[fil_num])) // reads each line in the file
  {
    if(strstr(line_p,place_name)!=NULL)  // looking for occurence of
                                                   // place in the line read
    {
          flag1=1;
          start_trans=strtok(line_p,",");   // extracting the transition.

/* The following for loop extracts the transition number */

       for(j=0;j<(strlen(start_trans)-1);j++)
                start_trans[j]=start_trans[j+1];
       start_trans[j]='\0';
       start_trans_num=atoi(start_trans); // converting the number to
                                                   // integer
       trans_list[trans_ctr]=start_trans_num; // storing in the trans_list
       trans_ctr++;
    } // end of if which finds the corresponding line with the given place number
    fscanf(fptr[fil_num],"%s",line_p);
  } // end of while which scans for the lines with the given place
  rewind(fptr[fil_num]);
  set_place=1;
  j=0;

/* The following while loop traverses the whole list of transitions to see
   if they are set. If they are all set to '1' the place can be enabled */
  while( ( (j==0)||(trans_list[j]!='\0') )&&(flag1==1)&&(set_place==1) )
  {
    if (t[fil_num][trans_list[j]]==0)
    {
      set_place=0;
    }
    j++;
  } // end of while loop which checks if the place can be enabled
  if ((set_place==1)&&(flag1==1))
    return 1;
  else
    return 0;
}

/*  The following function takes the place number (that is enabled) and
    file number (for attack pattern) as input and returns the list of
    transitions (in an integer array, transtn_list[]) that has to be
    checked if they can be enabled.  */

 void check_trans_list (int b, int fi_num)
 {
  int k,test_trans_num,t_val='t';
  char p_name[10],test_trans[10];
  char line_p_t[max_len];
  int trans_ctr=0;
  rewind(fptr[fi_num]);

  for(k=0;k<max_transitions;k++)
      transtn_list[k]='\0';  // Initialization
  sprintf(p_name,"%c%d%c",'p',b,',');
  fscanf(fptr[fi_num],"%s",line_p_t); // reads a line from the file

  while(!feof(fptr[fi_num]))  // loops until end of file
  {

/* The following if checks if the place name is present in the line read.
   Note that delimiter has to be included in place name to get the right
   result */

    if(strstr(line_p_t,p_name)!=NULL)
    {
       strcpy(test_trans,strrchr(line_p_t,t_val)); //getting the
                                                          // transition
       for(k=0;k<(strlen(test_trans)-1);k++)
              test_trans[k]=test_trans[k+1];
       test_trans[k]='\0';
       test_trans_num=atoi(test_trans); // getting the number of the
                                                   // transition
/* The following while and if constructs checks that the transition number
   is not already present in the transtn_list[] before adding a new
   transition */

           k=0;
           while((transtn_list[k]!=test_trans_num)&&(transtn_list[k]!='\0'))
       {
         k++;
       }
       if(transtn_list[k]!=test_trans_num)
       {
       transtn_list[trans_ctr]=test_trans_num;
       trans_ctr++;
       }
    } // end of if which finds the line with the given place number
    fscanf(fptr[fi_num],"%s",line_p_t);
  } // end of while which scans for the lines with the given place
 rewind(fptr[fi_num]);
}

/* The following function reads the transition number and file number(for
   attack pattern) and detremines whether that particular transition can
  be enabled. The function return zero if the transition can not be
  enabled and if the transition can be enabled, the function returns the
  transition number that can be enabled. */

int enable_trans (int c, int f_num)
{
 int t_val='t',k,set_trans,place_num,places_list[max_places];
 int places_ctr=0;
 char end_transtn[10],test_transtn[10];
 char *str1,line_t[max_len];

 rewind(fptr[f_num]);
 for(k=0;k<max_p_val[f_num];k++)  // Initilaization
      places_list[k]='\0';
 sprintf(test_transtn,"%c%d",'t',c); // transition needs to be checked
 fscanf(fptr[f_num],"%s",line_t); // reads a line from file
 while(!feof(fptr[f_num]))        // loops until the end of file
 {
   strcpy(end_transtn,strrchr(line_t,t_val)); // checks for the last
                                              // transition in the line
   if(strcmp(end_transtn,test_transtn)==0)   // checks if they are same
   {

/* The following part collects all the place numbers (before the
  transition number that needs to be checked). All these places have to
  be enabled for the transition we are checking to be enabled */

           str1=strtok(line_t,",");
        while(str1!=NULL)  // breaking the line into segments
        {
         if(str1[0]=='p') // looking for places
         {
           for(k=0;k<(strlen(str1)-1);k++)
                  str1[k]=str1[k+1];
           str1[k]='\0';
           place_num=atoi(str1); // getting the place number from place

/* The following while and if constructs checks that the place number
  is not already present in the places_list[] before adding a new
  place number */

           k=0;
           while((places_list[k]!=place_num)&&(places_list[k]!='\0'))
                {
                 k++;
                }
           if(places_list[k]!=place_num)
                {
             places_list[places_ctr]=place_num;
             places_ctr++;
                } // end of if which adds a new place into places_list
         } // end of if which looks for places in a line
         str1=strtok(NULL,",");
        }  // end of while that scans all the places in a line
```

```
    } // end of if which finds an occurence of test_trans at the end of
      //a given line

    fscanf(fptr[f_num],"%s",line_t);
  } // end of while that reads a line from input file at a time
  rewind(fptr[f_num]);

  set_trans=1;

/* The following while and if constructs checks if any of the places are
   not enabled. If any of the places is not yet enabled, the transition we
   are checking for can not be enabled */

  k=0;
  while(places_list[k]!='\0')
  {
    if ((p[f_num][places_list[k]]==0)&&(set_trans==1))
      set_trans=0;
    k++;
  } // end of while loop which checks if the place can be enabled
  if (set_trans==1)
    return c;  // returns the transition number that can be enabled
  else
    return 0;  // transition can not be enabled
}

/* The following function reads the file number (storing attack pattern)
   and returns the maximum place number and maximum transition number
   found in integer arrays. ( max_p_val[] and max_t_val[] ) */

int file_max_p_t(int file_num)
{
 int l,t_value='t',p_value='p';
 char inp_line[max_len],c_inp_line[max_len],max_p[10];
 char max_t[10],file_name[20];

 sprintf(file_name,"%s%d%s","pt",(file_num+1),".dat");

 if((fptr[file_num]=fopen(file_name,"r"))==NULL) // opening file
    printf("file %s could not be opened for reading\n",file_name);
 else
 {
   fscanf(fptr[file_num],"%s",inp_line);

   while(!feof(fptr[file_num])) //loops until the end of file
   {
         strcpy(c_inp_line,inp_line);
     fscanf(fptr[file_num],"%s",inp_line);
   }
   rewind(fptr[file_num]);

/* The maximum place number and maximum transition number is obviously
   present in the last line of the file. c_inp_line[], now has the last
  line in the file. */

    strcpy(max_p,strrchr(c_inp_line,p_value)); //extracting max. place
    for(l=0;l<(strlen(max_p)-1);l++)
           max_p[l]=max_p[l+1];
    max_p[l]='\0';
    max_p_val[file_num]=atoi(max_p);  // getting max place number
    strcpy(max_t,strrchr(c_inp_line,t_value)); // extracting max transition
    for(l=0;l<(strlen(max_t)-1);l++)
           max_t[l]=max_t[l+1];
    max_t[l]='\0';
    max_t_val[file_num]=atoi(max_t); // get max transition number
  }
  return 0;
}

/* The following function takes the command number and file pointer number
   (for output to be written) as input parameters. Then it searches the
   com_user file to see who isssued that particular command and then
   prints the user information along with the command number to output
   file using the file pointer number passed */

void print_user_info(int d, int e)
{
  int flg1=0;
  char user_info[50],c_user_info[50],comm[10];
  rewind(com_user_fptr);
  sprintf(comm,"%d%c",d,',');
  fscanf(com_user_fptr,"%s",user_info); // reading a line from file
  while((!feof(com_user_fptr))&&(flg1==0)) // searching the file until
  {                                          // match is found
    strcpy(c_user_info,user_info);
    if(strstr(user_info,comm)!=NULL)      // checking for match
    {
     sprintf(user_info,"%s%s","Command",c_user_info);
        fprintf(patt_fptr[e],"%s\n",user_info); // writing to output file
        flg1=1;
    }
    fscanf(com_user_fptr,"%s",user_info);  // reading a line from file
  }
}
```

Input Files Used in the Program

pt1.dat:
```
t0,p1,p2,t1
t0,p4,t3
t1,p3,t2
t2,p5,p6,p7,t4
t3,p7,t4
t4,p8,t5
```

pt2.dat:
```
t0,p1,p2,t1
t0,p3,t3
t0,p4,t4
t1,p5,p6,t2
t2,p7,t3
t3,p8,t5
t4,p9,t6
t5,p9,t6
```

pt3.dat:
```
t0,p1,p2,t1
t0,p3,t2
t1,p4,t3
t2,p5,t4
t2,p6,t5
t5,p7,t6
t3,p8,t7
t4,p8,t7
t6,p8,t7
t7,p9,p10,p11,p12,p13,p14,t8
t8,p15,t9
```

pt4.dat:
```
t0,p1,t1
t0,p3,t3
t1,p2,t2
t1,p2,t4
t2,p4,t3
t3,p5,t4
t4,p6,t5
t5,p7,p8,t6
t5,p8,t8
t6,p9,t7
t7,p10,t8
t8,p11,t9
t9,p12,p13,t10
t10,p14,t11
t11,p15,t12
```

pt5.dat:
```
t0,p1,t1
t0,p5,t2
t1,p2,p3,p4,t2
```

```
t2,p6,p7,t3
t3,p8,t4
```

pt6.dat:

```
t0,p1,t1
t0,p3,t3
t1,p2,t2
t1,p2,t3
t2,p4,t4
t3,p5,p6,t4
t4,p7,t5
```

inp_data: (sequence of command numbers passed)

```
322 3 7 20 4 5 23 6 2 45 6 9 53 1 4 2
11 99 10 9 70 3 2 47 4 5 32 9 7 8 12 12
7 75 3 42 2 6 29 7 5 8 6 24 8 9 12 9 13
14 11 10 9 90 10 11 14
```

(produces a combination of completed and developing attack patterns)

inp_data: (another sequence of command numbers passed)

```
322 3 7 20 4 5 23 6 2 45 6 9 53 1 4 2
11 99 10 9 70 3 2 47 4 5 32 9 8 12 12
75 3 42 2 6 29 5 8 6 24 8 9 12 9 13
14 11 10 9 90 10 11 14
```

(does not produce any completed attack patterns but some developing patterns)

com_user:(contains command numbers and user id who issued that command)

```
1,user23
2,user22
3,user5
4,user25
5,user30
6,user3
7,user11
8,user8
9,user45
10,user53
11,user32
12,user99
13,user47
14,user75
15,user50
16,user29
17,user66
18,user90
19,user16
20,user21
```

ALGORITHMS FOR DATA INTENSIVE KNOWLEDGE DISCOVERY

Dr. J. L. Potter
Professor
Department of Mathematics and Computer Science

Kent State University,
Kent, OH 44242

Final Report for:
AFSOR SUMMER RESEARCH EXTENSION PROGRAM

Sponsored by:
Air Force Office of Scientific Research
and
AFRL, Rome, NY

December 1999

ALGORITHMS FOR DATA INTENSIVE KNOWLEDGE DISCOVERY

Jerry Potter
Professor
Department of Mathematics and Computer Science
Kent State University
Kent, Ohio 44242
potter@mcs.kent.edu

Abstract

During the 1998 Summer AFSOR program, potential future Air Force problems were analyzed to determine their common features. The solution to many of these problems depends on an advanced database/knowledge base capability. Current trends imply that conventional larger, faster knowledge bases will be available from commercial off-the-self sources. However, in addition to conventional data/knowledge bases, the Air Force problems require knowledge discovery in raw unstructured data, use of semi-structured knowledge, integration of diverse knowledge sources, utilization of the knowledge in knowledge bases by non-technical users, access to data using "conventional communication skills", access to knowledge over long distances and from multiple sources. Several current hardware and software technologies were reviewed to determine their appropriateness for addressing these issues. As a result of this review, it is concluded that associative algorithms augmented with neural network techniques are well suited for time critical knowledge discovery. The research focused on two key components - knowledge extraction from raw unstructured and semi-structured text and associative databases. Neural networks are a critical element in both of these components.

ALGORITHMS FOR DATA INTENSIVE KNOWLEDGE DISCOVERY

Jerry Potter

1.0 INTRODUCTION

The Air Force today must deal with challenging problems that need fast, efficient knowledge discovery and analysis of vast amounts of structured and unstructured data. This data needs to be available for such applications as defense information warfare, automated information processing from sensor to shooter, global access to intelligence information, the ability of expeditionary forces to pick up and go any where at any time, the prevention of terrorism through knowledge discovery, using knowledge to provide the ability to react to biological warfare threats and real time planning [II98][1]. With today's sophisticated data gathering technologies, more data is being collected than ever before. However, it is impossible to extract information from all the data gathered in a timely fashion. The result is a complex collection of knowledge, information, structured and unstructured data. Methods must be found to sort through all of this data, extract facts and organize them so that the personnel facing the problem who makes the decisions, whether a database expert or not, whether in an office, on a plane, or in the field has access to the information they need to do their job. This research furthers the activities of the Intelligent Knowledge Base System Working Group of the Information Institute Workshop 98. The purpose of this research was to identify the best approaches to discovering, accessing and supplying the diverse and remote knowledge and information needed to solve today's and tomorrow's challenging situations.

The rest of this report is organized as follows. The next section is a brief review of data and knowledge bases, knowledge discovery, and hardware. The third section describes an overall design for the Air Force challenge problems. It examines some of the critical issues in detail and identifies what algorithms were chosen for additional study. The fourth section discusses the results of the study. The fifth section is the conclusion. References follow.

2.0 BACKGROUND

The advent of ever faster computers has created problems as well as solving them. In particular, more data is collected today than ever before. There are three problems. One is the overwhelming amount of data. There is more data in data and knowledge bases than can possibly be understood. There are orders of magnitude more "raw" data waiting to be processed. Second, the data and knowledge is not necessarily easy to access logically or physically. Logically, it is often in a highly technical form not readily accessible to non-specialist personnel. Physically even given today's world wide access to networks, remote field terminals may have restricted capacity and limited computing power. Third, traditional architectures are becoming a hindrance to improved performance via greater chip density.

Modern database technology has attacked the abundance of data using various techniques such as active agents and "push" technology whereby data is processed in the background and relevant information is "pushed" to your attention. "Pull" techniques go after the data requested by the user. Databases have evolved into knowledge bases where active components can filter and search data with the efficiency of a librarian. Yet these techniques are not enough. Knowledge discovery in unstructured data in the form of newspapers, telephone conversations, TV, photos, raw electronic transmissions, etc. is most likely to be the source of the most current information.

Research into knowledge discovery has been on going for many years. For example, in the early 90's, the Message Understanding Conferences [MUC1-6] sponsored by NRaD and DARPA focused on analyzing free text, identifying events of a specified type and filling in data base templates. An alternative activity uses automatic program understanding [Kozaczynski, 1992, p. 217][2] as a paradigm for "data/knowledge" discovery in that programs are known to contain knowledge, yet they can be very difficult to understand. Typical approaches use syntactic knowledge of the domain to build a model of the abstract concepts to be recognized, starting with a small model and evolving it as the domain becomes better understood. For example, in programming, the "read data, process data" sequence is almost universal. The fact that the data must be read before it is processed can be specified as a syntactic constraint. Facts and relationships such as this can be incorporated into knowledge based rules.

AF problems have special demands on the hardware. In particular, more raw data exists than can possible be processed using COTS hardware. State of the art computing power is needed. Furthermore the field deployable units need as much computing power as can be packaged in them since they are dealing with a time critical situation in which the user must dynamically generate and access data/knowledge bases unique to the situation.

Parallel computing was extensively explored in earlier years as a solution to the computing needs of intractable problems. The hypothesis was that at some point, large scale integration would reach an upper bound beyond which it would be physically impossible to improve and then parallelism would take over to allow large numbers of chips to be grouped together into the needed super computers. However, chip capacities have continued to improve. As individual computers have become more powerful and operating systems more sophisticated, it is felt that explicit parallel machines are not necessary except for very special applications and that networks of computers could be used instead of large parallel machines for most other applications.

However, the problem today is that large scale integration is so successful that engineers are having difficulty determining what the most effective use of the available real estate is. [Brockman, 1997] says that "classical architectures are already reaching their limits ...[p. 2]".[3] One of the critical barriers to improved performance of conventional architectures is the "gap between the needs of a modern CPU core and the capabilities of the memory system." [Keeton, 1997, p.2][4] - the I/O bus bottleneck. "An alternative approach to system organization, that promises to deliver high performance with the greatest silicon efficiency, is to place the processing logic on the same die as high density memory, especially DRAM [Brockman, 1997, p. 3]." Data parallelism is needed to make the most efficient use of chip real estate in the Processor in Memory (PIM) approach and programming techniques such as associative programming are needed to make the additional computing power available to all.

3.0 KNOWLEDGE BASE GENERATION, ACCESS AND UTILIZATION

The goal of this research was to determine where advancements in state of the art software and hardware technology would be useful to help solve Air Force data/knowledge base generation and utilization for grand challenge problems. The first step was to develop a broad framework common to the AFRL Challenge problems. Specifically all of the problems have a need for data and knowledge. They need access to existing data/knowledge bases and the information in raw, unstructured text. Access to the unstructured text requires that the knowledge/data be extracted and organized into useable form.

A second theme common to many but not all AFRL challenge problems is the need for a field deployable unit which has a direct link to the fore mentioned knowledge/data. These two components called the base anchor system and the SKBAM and their relationship is illustrated in Figure 3.0-1.

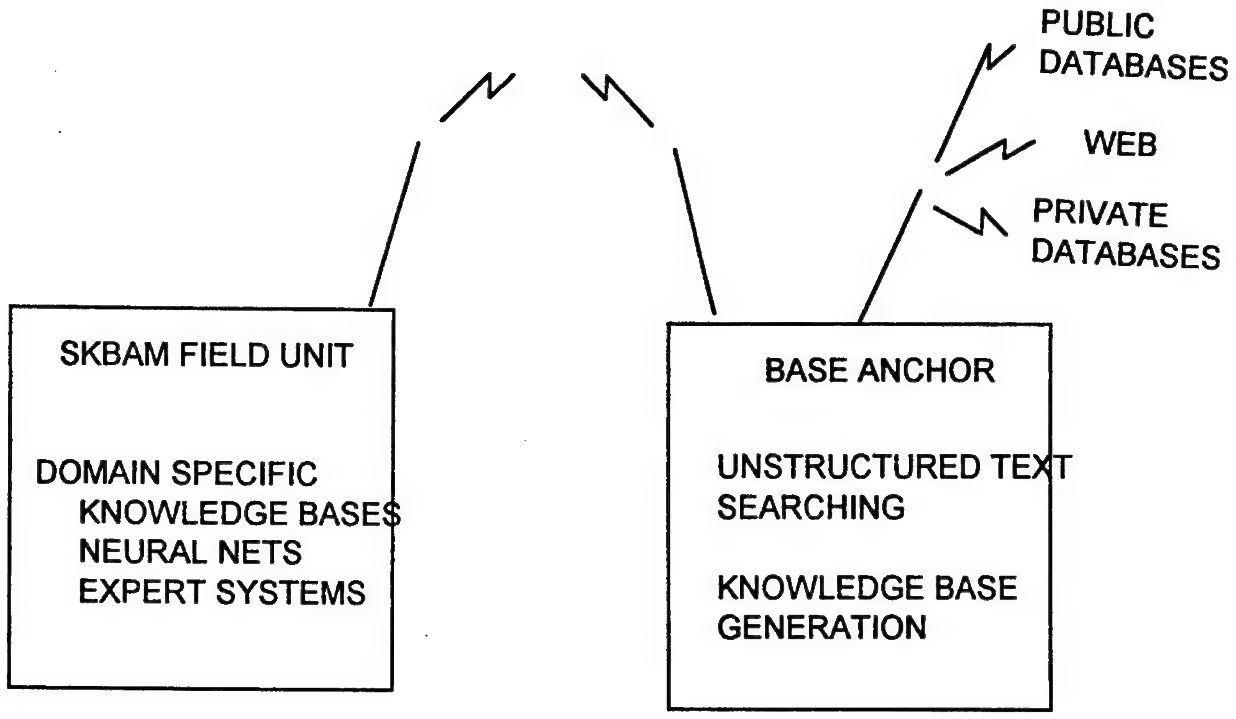


Figure 3.0-1 -The SKBAM and Anchor

The anchor system is responsible for 1) the on-going background processing of raw data, and knowledge base generation, 2) the on demand, real time support of the SKBAM field deployable units. It provides the link to the "outside" world of public and commercial data bases as well as military data bases and to the domain specific data/knowledge bases generated by the background processing.

The 1998 Information Institute Minnowbrook conference (II98) identified several technology shortfalls needed by the proposed AFRL challenge problems. In the Processor category, Very Large Data/Knowledge Bases, Parallelization, Contextual Understanding, Special Purpose Architectures and Smart Chips were identified as areas of technology shortfall. The technology shortfalls in the intelligent software include data capture, formal methods, knowledge discovery, learning, information maintenance, mobile computing, mediation, inference and ontology. Since the challenge problems are not mass market issues, it is unlikely that COTS equipment, hardware or software, will be developed to address these shortfalls in a manner consistent with the challenge problems. Therefore several were selected for additional analysis.

Considerably synergy can be obtained if the shortfalls from the intelligent software and processor categories are addressed together. For example, many of the items from both categories such as very large data/knowledge bases, parallelization, mobile computing, formal methods, learning, knowledge discovery, ontology, contextual understanding and smart chips are addressed in the development of a smart portable field deployable terminal (a Smart Knowledge Base Access Machine - SKBAM) that works closely with the nondeployable base anchor.

Since the needs of the SKBAM may impact the design of the anchor system, it is discussed first, then the anchor system itself, and finally the hardware support issue is addressed.

3.1 A Smart Knowledge Base Accessing Machine (SKBAM)

In those challenge problems were a field deployable unit is needed, it is anticipated that the problems would need the following requirements for a SKBAM:

1) Pertinent data bases and knowledge bases with domain specific expert systems and neural networks. Data from the selected sources will be customized for each challenge scenario and pre-loaded into the field deployable device as a local suite of data and knowledge

bases (SDKBs) fronted with a non-technical user friendly, interactive man machine interface requiring only conventional communication skills.

2) The field deployed user will verbally query the SDKBs. Simple queries will be answered directly. More complex queries will be referred to the appropriate knowledge bases in the SKBAM's anchor system as needed.

3) As the user queries the SDKBs interactively, new data from previously identified sources will be sought, down loaded and integrated into the local SDKBs using push technology by the domain specific anchor system.

In order to accomplish this functionality, the SKBAM's OS software would allow its local SDKBs to I) interact with global/remote knowledge and data bases, ii) "pre-load" its memory with anticipated background knowledge bases generated by the anchor system, iii) learn what kind of information is normally requested to help formulate queries, iv) learn to retain down loaded information so it does not have to be down loaded more than once, and v) to work with very large assemblages of incomplete knowledge and raw data to perform such tasks as knowledge discovery and integration.

Figure 3.0-1 is a symbolic functional illustration of a SKBAM in the field. The user is surrounded by intelligent tools which are capable of accessing a vast variety of local information and data and knowledge bases which are linked to the outside world via a communication link which no matter how big is small compared to the amount of data in the outside sources.

The SKBAM would be voice/speech enabled and would support dynamic man/machine exchanges. Consider for example, dealing with knowledge discovery in unstructured data. The domain specific SDKBs would suggest knowledge templates appropriate for the current situation. The user would accept and/or modify the suggestions and tell the SDKBs to proceed. The SDKBs would search their own data and make requests of the anchor system for domain specific or global data as needed. Based on what was found, the SDKBs and user would make new suggestions. This is a combination of push/pull technology. The pull part limits the processing to what the user of the SKBAM is immediately interested in. The push part informs the user what was found to be of interest based on the SDKB/user designed queries.

3.2 Knowledge Acquisition Anchor System

The techniques used for on-going background domain specific knowledge extraction is coordinated with that needed for the SKBAM device itself via the SKBAM anchor system. In the on-going background operation, raw data is processed looking for pertinent information. One goal of this research is to identify and develop techniques to efficiently process large amounts of raw text and populate knowledge and databases. One key to knowledge extraction is the realization that the text itself may be unorganized, but there is a very specific organization to the domain specific data knowledge being sought. That is, an expert designs a template database for the facts and knowledge sought. Raw text is searched and when items specified in the template are recognized they are put into the knowledge base.

Experts design the anchor resident domain specific data/knowledge bases and knowledge templates. It is assumed that in general there are many experts and many separate domain specific tables. Given the vast amount of data to be processed, the extraction process must be as efficient as possible. So all templates are processed simultaneously. However, in order to simplify the details, this discussion is presented as if there were only one set of tables and templates.

In all of the Air Force Challenge problem domains, the standard "W" questions - who, what, when, where, why and how - are very important. The primary difference is in the vocabulary of the different domains. Thus the vocabulary for terrorism may contain such words as Waco, Branch Davidians, FBI, tear gas, etc.. While the vocabulary for national nuclear security would consist of nuclear, test, antiballistic missile, etc.. The domain expert enters the words in the appropriate categories as shown in Figure 3.2-1. The ANGSE search engine, described next, then populates the tables.

3.2.1 The Associative Net Guided Search Engine (ANGSE)

The basic approach to associative fact extraction is to search for combinations of specific keywords. These associations are context/syntax free. There presence in the same sentence, paragraph, section or "article" is sufficient for extraction. These combinations of words are then passed to an array of neural nets whose function is to determine 1) the significance of the facts, 2) the expert system which should be notified, and 3) which expert agents, if any, should be activated

WHAT	WHO	HOW
treaty	China	contrary
nuclear	terrorist	fired
ballistic	Russia	probe
radiation	army	test
test	congress	simulation
evidence	North Korea	launch

Figure 3.2-1 A Partial Table for National Nuclear Security

to extract additional information. The architecture of the ANGSE is shown in Figure 3.2.1-1. The major components are 1) the associative fact extraction component, 2) the neural net domain classifier, and 3) the domain knowledge extraction agents. Each component is described in detail in the following sections.

3.2.1.1 Associative Fact Extraction (AFE)

The AFE component is designed to process large amounts of raw text looking for domain specific vocabulary words. A set of sensitivity indicators can be set to establish how many hits per sentence, per paragraph, per section and per document must be present to cause entries to be made into the domain tables. The sensitivity indicators can be set as a lower bound or as an upper bound. The upper bound mode can be used during periods of high activity to filter out those documents that are dominating the domain and allows through those texts which contain more subtle inferences. In any case, it is assumed that the majority of entries generated by the AFE component are incomplete and that more analysis of the raw text is needed to fill in the blanks. This follow on analysis is initiated by the domain specific knowledge bases which are alerted to incoming facts by the neural net classifier.

3.2.1.2 The Neural Net Domain Classifiers (NNDC)

As each associative template record is composed it is presented to the neural net classifiers. Since many domain specific tables (DST) are being populated simultaneously, it is likely that an associative record should be passed to more than one table. The NNDC notifies the appropriate domain experts of the incoming information.

The NNDC is the key component connecting the operation of the field unit with the anchor background database generation component. It resides and is trained in the anchor system but it is also loaded in a domain specific SKBAM and is designed to modify its weights and update itself quickly to handle the dynamic events of the field environment.

The neural net is a 4 layer feed forward net with back propagation. The first 3 layers consist of a standard feed forward neural net architecture. The 4th layer transforms output activations from the 3rd layer to probabilistic output where Sum(p) = 1.0. The probability distribution is based on the Boltzman Distribution [Equation] where [alpha] controls the degree of randomness in the decision making process[5]. See Figure 3.2.1.2.

$$p(E \mid f) = \frac{e^{1/\alpha O(E|f)}}{\sum_{i=1}^{n} e^{1/\alpha O(E|f)}}$$

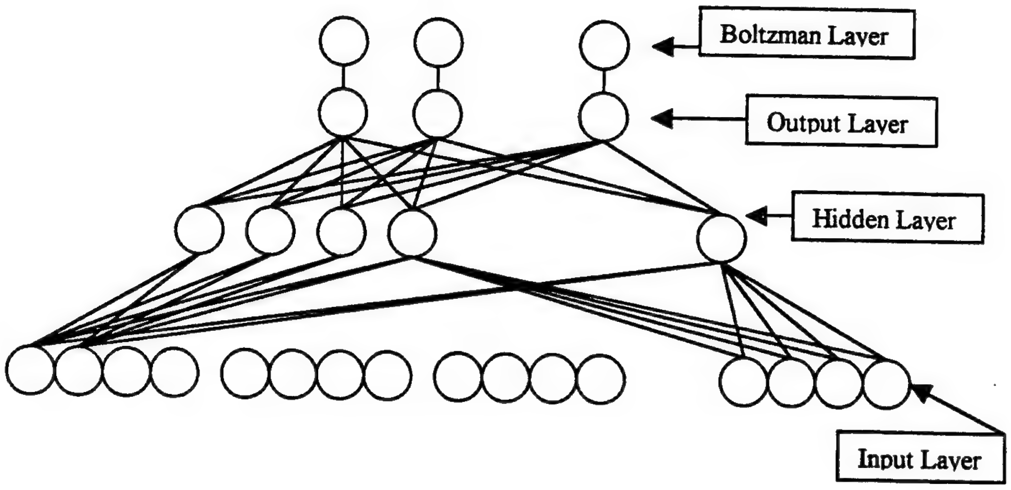


Figure 3.2.1.2-1 The Boltzman Distribution and Four Layers of Net

The input layer consists of groups of nodes representing categories of synonyms (facts) in the particular domain in question. For example, in the domain National Nuclear Security the word nuclear and all synonyms for nuclear would be assigned to four input nodes(to indicate its relative strength), the word missile and all synonyms for missile would be assigned to four input nodes, etc. The hidden layer would be initially assigned to be SQRT(M*N) where M is the number of input nodes and N is the number of output nodes[6]. This would then be adjusted dynamically to produce the most efficient minimization of error during training. The output layer and the Boltzman layer would consist of one node per expert system. The exact usage of the input layer and the output layer are described below.

The NNDC system as a whole goes through a 3 step process: training, loading and deployment. In the training and loading phases, there is much room to exploit data parallelism (SIMD) and possibly instruction parallelism (MIMD). Neural network performance can be significantly enhanced through SIMD parallelism, as implemented by [Rosenberg 1987][7]. MIMD and/or Multiple SIMD parallelism can be implemented across the expert systems so long as the problems of concurrent read and write to the database(s) are addressed.

3.2.1.2.1 NNDC Training

During the training phase, I/O vector pairs are presented to the network. Input to the network consists of categorical information for the domain in question. An output vector would consist of one expert system selected to be associated with the input vector. For example, if the domain is National Nuclear Secutity the following could be a list of input categories: nuclear, test, ballistic, radiation, third world, China, Russia, terrorist, and plutonium. A particular input vector presented to the network might be missile, test, and third world. That example input would be associated with a particular expert designed to deal with that sub-set of topics. Such an expert might be Missile Testing in Third World Countries. I/O pairs are selected from human experts and real world data. This is standard for this type of neural network model and training proceeds by presenting many (hundreds) I/O pairs to the network. One pass through the training set is an epoch, and many (thousands) epochs complete training.

Since I/O pairs are selected from human experts and real world data, some I/O pairs will conflict. This follows from real world expert decision making where two experts presented with the same information will select different methods of analysis. For example, in the previously given list (missile, test, and third world) most experts would choose to analyze this data in the context of Missile Testing in Third World Countries, although some experts may choose Arms Sales to Third World Countries or Super Powers Affiliated with Third World Countries. This "conflict" is not really a conflict since it follows from human decision making where small undetectable variations in data or personal biases and experience cause human experts to differ in opinion. These small variations that cause a difference in opinion in human experts would be tedious, if not impossible, to program into a rule-based system although the neural net captures the variation by directly learning from the expert's decisions rather than all the input to the expert's decisions. Note that the training set must reflect the more likely experts to deal with a set of inputs by repeating those I/O pairs several times in the set.

Training is concluded when there is no significant change in output activations (3rd layer) or error has been minimized as in standard back propagation. Error in this model can not be minimized to the same degree as would normally be found for a neural network that is learning a strict function since some I/O pairs will always be in error compared to other I/O pairs. When training is concluded, a set of inputs can be presented to the network which will produce probabilities that represent the most likely expert to handle the input, the second most likely expert to handle the input, etc.

During the training phase the 4th layer is unused. The inputs are assigned as four fully activated nodes per input fact. Initial experiments using a neural net in this fashion show that for a small M (24) and N (3) and a training set of around 100 I/O pairs, training can be completed in under 10,000 epochs.

3.2.1.2.2 NNDC Loading

Once the neural network is trained in a particular domain, it can be inserted in to the model for the loading phase. Assuming all other parts of the system are in place, including the (partially) filled domain specific database, the neural network is now used to select experts to process raw text. A document is first presented to the associative fact extractor. The output of this step is a (incomplete) table of who-what-where-when facts (See Section 3.2.1.1). This table is used in two ways. First, the table is "collapsed" so that the whole document can be presented to the neural network at once as a single vector. The collapsed table is a histogram of all the facts in the document along with the associated strength of the facts based on its frequency of occurrence. The frequency of occurrence for a particular fact determines how many of the four input nodes will be activated for that fact. Facts with the highest frequency are said to be "very frequent" and all four nodes are activated. Facts with the lowest frequency are said to be "very infrequent" and only one node is activated. Two nodes activate if the fact is "infrequent" and three nodes if the fact is "frequent." "Frequent" and "infrequent" are based on how often the fact appears between "very infrequent" and "very frequent."

The neural net selects the experts to deal with the facts in the document in question. The experts are selected with probabilities that determine the most likely expert to handle the document, the next most likely, etc. It is here that the 4th layer of the network is used to generate probabilities from the output activations. By varying (in the 4th layer, the degree of probability per expert and the number of experts selected can be adjusted. Once the experts are selected and ordered by probability, the table is presented to each expert line by line in it's original form (uncollapsed). For each line in the table the experts process the information by drawing from the database(s) to fill incomplete facts in the table and also insert facts to the database(s). Probability information can also be transferred to the database to aid in conflict resolution for inconsistent facts.

In this phase, the facts are presented to the network by activating one to four nodes per fact or zero nodes if the fact is absent. This allows the network to decide on the strength of the fact and it's importance to the document. Presumably, the number of times a fact (and all of it's synonyms and pronouns) are mentioned in a document the more it represents the topic of that document. In training, we are dealing with facts in isolation, so facts are presented full strength for the purpose of learning. In the loading phase, facts are presented to the network in the presence of other facts, so they are presented with frequency information. This allows the network to make probabilistic judgments as to which expert(s) should handle the document. For example, during training in the domain National Nuclear Security, the facts missile, test, and third world presented in isolation would highly activate the output to the sub-domain expert in Missile Testing in Third World Countries. During the loading phase, these same facts may be in the presence of other facts such as terrorist and plutonium. These other facts may have a higher or lower relative frequency. It is this frequency that will decide if the output for sub-domain expert in Missile Testing in Third World countries is highly activated compared to other experts which may be more appropriate to handle the additional information terrorist and plutonium.

During the loading phase, massive amounts of unstructured data can be inserted into the structured database. The experts themselves could interact to refine data before it is inserted into the database. Also, the probabilities generated by the network can be incorporated into the facts of the database so that more frequently used or more common facts are retrieved more quickly, similar to the way more common definitions are listed first for a dictionary entry.

3.2.1.2.3 Field Deployment

After training the network and loading the database, the neural net and the synonym matcher are transferred to the field agent. In the field, the user submits a query in the same form as the table entries from training and loading. The user may submit a who-what-where-when query with one or more empty facts. These facts are transferred to the neural net at full strength (as in training). The output of the neural net and the query is transmitted back to the base anchor system where the experts and the database are online. The experts are assigned based on the neural net probabilities transferred from the field agent. These experts read in the incomplete user information, query the data base, and return the completed information to the field agent.

For example, the field agent may be have a neural net and several experts trained in the domain National Nuclear Security. While in the field, the user may be interested in uncovering the possibility that nuclear material has been bought recently by a known terrorist in the Middle East. The query could be Who(Terrorist Name), What(purchase plutonium), Where(Middle East), When(?). The expert systems assigned by the field agent network would query the data base, interact with each other to refine the data, and return their results ordered by probability (as initially assigned by the neural network). The user would select the results(s) most suited to his query, and that choice would be used to update the local (field agent) network. In this way, the network inserted into the field agent would learn and adapt to the particular user submitting queries.

It is intended that this system will be to uncover facts with results similar to data mining. Much knowledge is not explicitly stated but can be uncovered by a combination of facts and documents. Also, the queries themselves can be taken as new facts to the experts. Maybe the experts don't know right now if nuclear material has been bought recently by a known terrorist in the Middle East. However, the question itself raises the fact that we should know if nuclear material has been bought recently by a known terrorist in the Middle East. This incomplete fact can be inserted into the database(s) just as would be done in the loading procedure. The next time a document(s) on that subject is read into the system that incomplete entry may be filled.

3.2.1.3 Domain Knowledge Extraction Agents (DKEA)

The third major component of the ANGSE are the domain knowledge extraction agents. Upon notification of a new relevant associative fact record, the domain expert analyzes it to determine its completeness. If components are missing, the domain expert notifies knowledge extraction agents which analyze the raw text in more detail to extract missing pertinent information. Due to limited funds and time, we were able only to develop a few agents. Three of them are 1) who, 2) when and 3) not.

Natural language makes extensive use of pronouns and pronoun references. The purpose of the "who/what" agent is to follow pronoun chains and resolve referents. For example, when the "who" agent is notified of the sentential source of an association. It searches that sentence for a noun-pronoun subject. If a pronoun is detected, the agent searches previous sentences for either a repetition of the same noun-pronoun, or its referent. For example, consider the keyword "compound" in the Waco domain[1]. It of course refers to the "Branch Davidian compound" near Waco, Texas. However, in most texts pertaining to Waco, the word "compound" is used as a pronoun for "Branch Davidian compound." This creates a problem in raw text searching since "compound" is not an unusual word and may appear in many raw text documents only a few of

[1]It should be noted that the domains selected for this research were selected randomly from the news events at the time this research was performed.

which refer to Waco. To address this issue, the pronoun agent must trace any records with "compound" in them back to the complete reference.

Unstructured raw text is of course a misnomer. The text is not organized into relations or tables but it is organized into sentences, paragraphs, sections and documents (or conversations). During the preparation of the raw text for analysis, every word is assigned a sentence, paragraph, section and document number as shown in Figure 3.2.1.3-1. Thus a pronoun such as "compound" can be traced back (or if necessary, forward) sentence by sentence and until is referent is found. This algorithm, described below, is well suited for a data parallel processor.

What	Who	How	Document Number	Section Number	Paragraph Number	Sentence Number
treaty	senators	testing	1	1	0001	0002
test	.	shoot	1	1	0002	0005
test	.	down	1	1	0002	0005
rocket	.	test	1	1	0002	0005

Figure 3.2.1.3-1 Text Structure Code

First the word of interest is found in all sentences. Next the article "the" is searched for. Every occurrence of "compound" proceeded immediately by "the" is deleted since "the compound" is the pronoun for which a referent is sought. The remaining occurrence of "the" and "compound" brackets the definition as shown in Figure 3.2.1.3-2. The definition is then extracted and broadcast to all pronouns so that the search need only be done once.

ATOM	YY,	ZZ,
on	0	0
the	1	0
Branch	0	1
Davidian	0	1
compound	0	1
near	0	0

Figure 3.2.1.3-2 The Bracketed Referent

A second example is the "not" agent. The polarity of every fact extracted must be determined absolutely. Since initial associations are determined without analyzing the entire sentence, every fact requires that the "not" agent be initiated. Since, in general, the associative search engine may extract many more "facts" than can be processed. The "not" agent is only fired when the neural net determines that the fact should be established. The not agent looks for the key word "not", the contractions "nt" and "n't" as in "can't", "won't." It also looks for other negative clues such as "un" prefixes. Thus it must be able to establish that "The judge was unhappy with the verdict" and "The judge was not happy with the verdict." both have negative polarity.

A third example is the "when" agent. It simply looks for keywords such as Monday, Tuesday, yesterday, tomorrow, week, month, and year. However, it must look for specific modifiers also such as "next week", "last week", etc. But the more difficult task is to associate a time with a specific association because frequently, the "when" information may be several sentences or even paragraphs away, and like the "who" agent, it must be able to establish and follow referential chains. And like the "who" situation, the time referents may be either in front of or after the referent.

3.2.2 Associative Databases

The SKBAM is a dynamic interactive interface on a portable terminal used by an emergency workers. The users may have no formal training in Data/knowledge base use (i.e. they do not know SQL). They may not be able to type well. They do not know how to build new relational databases. But it is critical for them to be able to build dynamic, minimally structured data bases that can be accessed by queries approximating natural language. Associative databases fit this need. They are an intermediate step between raw unstructured text and a highly structured relational data base. Associative databases search unsorted data directly and use Falkoff's algorithm[8] to find the "best" (maximal or minimum) values. Thus the data does not need to be sorted, inverted lists are not necessary and all variables (fields of data) are available.

For example, considerable a resource database. In a disaster situation, knowledge of the available resources is vital. In particular what professionals are close by - policemen, sheriffs, national guard, air force reserves, etc. What special expertise do they posses? Arms expert, explosive expert, evacuation expert, crowd control, dog handlers for tracking, dog handlers for finding victims? A database of physical resources such as cars, trucks, fire engines, ambulances, hospitals, etc. is also important. It is assumed that databases of this nature have been compiled ahead of time, but obviously in a real dynamic emergency it is out of date when it is needed. That is, some of the personnel are on vacation, one or more vehicles is undergoing repairs, etc. Therefore it is critical that this data base can be updated easily and efficiently by a non database expert. Associative data bases seem to be well suited for this job.

The relational database technology and the knowledge base techniques of today are outstanding. But conventional knowledge base technology is too structured. It is unlikely that the "syntax and semantics" of a specific terrorist activity such as that in East Africa can be specified in detail in advance. Unfortunately, the current research trend into object oriented databases is heading in the wrong direction for SKBAM applications since "Object data models are very rigid and inflexible ...[Burleson, 1998, p.3]"[9] A more interactive, informal, associative approach is needed.

Associative search and databases have been researched and discussed for many years, and even though they are still primitive compared to relational techniques, they are ideal for the remote, real time interactive processing environment of SKBAM. Associative searching is a depth first search process as opposed to the relational database breadth first. As a result, associative searching is faster in situations where a first answer is important.

Several languages, such as ASC[10][Potter, 1992], have been designed based on associative principles. The main advantage of these techniques over conventional database languages is that they are not only capable of processing unsorted database data, but also "unstructured" data such as raw text and imagery. Associative languages such as ACE[11] have a natural language like syntax, not a mathematical set based syntax like SQL and thus are easier for non-database specialists to learn and use and they are suitable for voice input.

Associative techniques such as structure codes and associative unification can be very useful when dealing with knowledge discovery and large amounts of unstructured data. A structure code is a tag containing structural information which is attached to each datum. The document number, section number, paragraph number and sentence number of Figure 3.2.1.3-1 is a structure code. The advantage of structure codes is that data does not need to be physically reorganized. The datum is static in memory, structure is added by modifying the code as processing progresses. Structure codes can be used to learn at the software and hardware levels. Table 3.2-1 illustrates some of the similarities and differences between associative and relational databases.

	Associative	Relational
Basic Data Organization	tabular	tabular
Key Fields	not required	required
Associative access	yes	no
Languages	natural language based (ACE)	math based (SQL)
Row access	optionally directly named	requires search operation
Column (field) access	names not required	names required
Knowledge of table structure	not required	required
Duplicate entries	allowed	not allowed

Table 3.2 - 1 Comparison of Associative and Relational Databases

3.3 Hardware Considerations

The selection of an architecture for the SKBAM and anchor base system is critical. Multi processor (MP) designs (including MIMDs and clusters) are the "default" parallel computer. However, in light of the extensive dynamic data parallel computational requirements, the utility of associative processors (APs) are also investigated. See [Potter, 1992] for a detailed description of associative processors.

In traditional complexity analysis a basic operation such as a compare is often used to measure the complexity of an algorithm. If the basic operation is common to different architectures, then the "power" of different algorithms which execute on these machines can be compared. Thus conventional sequential computers can be compared with machines that have numerous processors, networks of processors and even phantom machines such as the PRAMs. as long as the analyses are "normalized" by the number of CPUs. Unfortunately, data parallel and associative processor are mis-analyzed by this standard comparative analysis technique. The following items illuminate the problems and examine the claim that APs are better than MPs in general for massive data parallel tasks and for the SKBAM in particular.

3.3.1 Algorithm Conversion

The attitude that MPs are "better" than APs is promoted by misleading examples in introductory computer science texts[12] of how easy it is to modify "embarrassingly parallel" sequential algorithms such as the Manhattan telephone book problem to run on an MP architecture . Many sequential algorithms are erroneously analyzed when ported to MPs because the induced communication component of the MP implementation is often ignored. Most programmers simply do not understand that when they convert a sequential algorithm into an MP algorithm, that they are introducing a potential NP hard synchronization /communication component which dominates the converted parallel algorithm.

A commonly accepted relationship for the increase in speedup for problems is:

speedup = Ts/Tp

where: Ts is the time for the sequential algorithm and Tp is the time for the parallel algorithm. The error, taught in a freshman text, is that the time for the parallel algorithm can be calculated from the time for the sequential algorithm by the formula:

Tp = C(Ts/U)

where U is the number of CPUs. This formula assumes that the additional complexity introduced when the sequential algorithm is adapted to the MP algorithm can be accounted for by some constant C, and that the speed up is linear.

This is a major misunderstanding of the nature of control parallelism. Notice that when converting the sequential pseudo code of Figure 3.3.1-1a into the MP pseudo code, four additional lines of code are needed for communication and synchronization - an INTRODUCED COMPONENT not in the original sequential algorithm and one that is independent of n, the size of the Manhattan telephone directory.

```
for (i=0; i<n; i++)
  if (searchName == dataName[i])
    return phoneNumber[i];
```

a - The Sequential Solution

```
synchronize
send the searchName to U processors;
for(i=0; i<n/U; i++)
  if(searchName == dataName[i])
    return phoneNumber[i];
synchronize
reduce answer
```

b - The MP Solution

Figure 3.3.1-1 - The Manhattan Telephone Book Problem

The communication component introduces a real time scheduling aspect which CAN NOT BE IGNORED. Consider the code in Figure 3.3.1-2, which represents the "ultimate" in (data) parallelism with one processor for each data item to be searched (i.e. U=n). In this limit, the

```
synchronize
send the search name to U processors;
if (SearchName == data name)
   return phoneNumber;
synchronize
reduce answer
```

Figure 3.3.1-2 - The "Embarrassing Parallel" Algorithm

synchronization effort is not "small" or "constant" and indeed can not be ignored since as the time for searching becomes smaller by adding more processors, the time for communication BECOMES LARGER and dominates the complexity analysis of the parallel algorithm. This fundamental misunderstanding of MP algorithms is the basis for much of the frustration that occurs when scientists parallelize their code and instead of obtaining a linear increase in through put actually achieve diminishing returns per processor added until achieving a negative increase at the point that adding a processor actually requires more communication overhead than the processor can contribute to solving the problem. In contrast, since an AP has only one instruction stream the problem can be solved in constant time if there is one PE per data item.

3.3.2 CPUs and PEs are not equivalent.

The standard practice of coupling of the ALU and instruction stream (IS) components of a computer into a CPU is a major cause of inaccurate AP versus MP comparative analysis. In particular, the traditional emphasis on the effort of the ALU (i.e. the data parallel component) at the expense of ignoring the impact of the IS (the control parallel component) allows the perception that sequential algorithms can be implemented on MPs simply by replicating the code. In order to more accurately compare algorithms between different architectures, MPs, APs and sequential, it is essential that the CPU be divided into IS and ALU components. In the case of sequential algorithm analysis the result is the equivalent as traditional analysis. However, in the case of MP algorithms, the importance of the communication component (IS) becomes apparent as described in Section 3.3.1. And in the case of APs, the algorithms are not falsely penalized for having large numbers of "phantom" ISs. Section 3.3.3 below expands on the implication.

3.3.3 Is a SIMD a Parallel Processor?

In Flynn's taxonomy of MIMD, SIMD, SISD and MISD, MIMDs and SIMDs have traditionally been grouped together as parallel processors. However, as discussed in Section 3.3.2, if the CPU is divided into an ALU and IS, then SIMDs should be grouped with SISDs. That is, for comparative analysis, the grouping based on ISs (i.e. the control parallel part) is more critical than the grouping based on ALUs (the data parallel part). Certainly a SIMD is a parallel processor, but of the two components, ALU and IS, the IS component is by far the more powerful (and expensive). That is, the IS embodies the Turing machine/FSA capability central to the power of a general purpose digital computer. The ALU by itself can only process two numbers. When dealing with different algorithms on similar architectures and CPUs, distinguishing between these two components is not critical, but the extension of this approach to different algorithms on different architectures leads to inaccurate comparisons.

For example, commonly, an m processor MP and an m processor AP would be considered to be of similar power. But this is equating m IS/ALUs against only one IS with m ALUs (weak ones at that), surely not a fair equivalence even in the world of comparatively analysis where constant factors can be ignored. But order of magnitude factors must be accounted for. It is easy to understand that the power of an IS is as least an order of magnitude more powerful than that of an ALU. Thus it is more accurate to treat APs as SISDs with very wide data paths than as the equivalent of control parallel computers.

As another example of the misunderstanding, consider that many feel that it has been proven that broadcasting is a log n operation for all parallel computers. In reality, any parallel computer with multiple instruction streams does take log n time. But since an AP only has one IS it can broadcast in constant. This is easy to prove - the data to be broadcast takes the same path as the instructions themselves. If you accept that any operation in a SIMD is constant time, compare, add, etc., then you must accept that broadcast is also.

3.3.4 Wide Memory to ALU Bandwidth

One of the simplest measures of an efficient processor is the amount of data that can be delivered for processing in each cycle. In contrast to MPs where instructions and data share a common path between memory and the ALU, in an AP the paths are different. In a single MP, this pathway may be 32, 64 even 128 bits wide - in an effort to improve the data delivery ratio. Yet in the inner most nested loop shown in Figure 3.3.4-1a, five instructions are required to process one datum (a five to one bits processed ratio). In an AP, the memory to IS path may be the same size as in an MP, but the data path can be thousands of bits wide. With a modest 512 bit wide data path as in Figure 3.3.4-1b, only eight instructions are required to process 512 bits (a one to two bits processed ratio). For this simple example, the AP is 10 times more efficient than a single MP. The comparison improves for the AP when additional MPs are added because of the additional communication overhead MPs require.

3.3.5 Keeping the Processors Busy

In MPs, processors (CPUs) are a major resource and must be efficiently utilized, in APs, however, PEs are frequently only one bit wide, consisting of a very small number of gates and it is often more efficient to "waste" it than to execute the extra instructions necessary to make use of it. This "disadvantage" of APs occurs when conditionals are present in a program, since the AP must execute all branches sequentially. This "disadvantage" frequently results in long sequences of uninterrupted code allowing very efficient programming and data pipelining. The inner most loop of a program is executed in parallel in the hardware - the essence of data parallelism. There are cases with long stretches of conditional code, where executing the alternatives in parallel is cost effective. In these situations, new AP designs such as MASC[13] with multiple ISs are capable of supporting these situations efficiently.

Some designs try to use the wider memory-to-cpu busses by fetching 2,4 or 8 instructions at one time. These "wide word" (or Very Long Instruction Word - VLIW) designs, however, suffer enormously from the basic programming limitations placed on them by the von Neuman model. That is, in standard sequential programming, processing varied data is accomplished by branching around the exceptions. In normal operations, branches are executed every 3 or 4 instructions. A branch is a discontinuity in program flow. Today's architectures have been modified with layers of cache memories and pipelines work but when instructions and data flow continuously and smoothly. The wide word technology depends on software - compilers and profilers to "fix" the branching problem. This is a poor solution, the best software can not make up for a poor hardware design.

```
for (i=0; i<512; i++)
  if (a[i] > 50)
    b[i] = a[i] * 0.33;
 else
   b[i] = a[i] * 0.5;
```

3.3.4-1a - MP Loop

```
if (a[$] > 50 )
  b[$] = a[$] * 0.33;
else
  b[$] = a[$] * 0.5;
```

3.3.4-1b - An AP Loop

Figure 3.3.4-1 - Loop Processing - MP vs AP

3.3.5 Bigger Computers are Smaller Computers

Increased computing power is achieved to a large degree by denser chips. This introduces the problem of "excess chip capacity." That is, when processor chips are able to include many capabilities beyond the basic 32 bit IS/CPU and when memory chips can store billions of bits of data with room left over, what should the extra space be used for? The primary bottleneck in today's architecture designs is the von Neuman bottleneck from memory to CPU. As described in Section 3.3.4 the AP is easily expandable to take advantage of any increase in size of the data path. Indeed, the APs architecture is ideal for the processor in memory, (i.e. PIM) approach.

[Brockman, 1997] claims that PIM is the only way to overcome the I/O bus bottleneck and discusses the spectrum of PIM designs. [Bowman, 1997] concludes "Existing architectures, whether simple, superscalar or out-of-order, are unable to exploit IRAM's increased memory bandwidth and decreased memory latency to achieve significant performance benefits [abstract]."[14] The most promising design for data/knowledge base applications are SIMD based associative computers such as the STARAN developed at RADC for database [Liuzzi, 1980][15] and other applications [DeFiore, 1971][16], [Stillman, 1971][17] and [Orlando, 1971][18].

3.3.6 APs are Easier to Program

There are three reasons why APs are easier to program than MPs. First, no inner-most loops as illustrated in Figures 3.3.4-1. Fewer loops means less code to write and debug. Second, less communication as mentioned in Section 3.3.1. Since there is only one IS, there is no inter-IS control communication (control parallelism). This is the component that most programmers identify as the "hard" part of parallel programming. The third reason is that the VACE [Potter, 1997] paradigm allows APs to be efficiently programmed by voice in a manner similar to a spreadsheet. While the paradigm can be implemented on any general purpose computer, it is a natural "match" to the AP architecture allowing a very efficient implementation of large data parallel database algorithms.

4.0 Results

Associative Fact Extraction, Neural Net Domain Classification and Domain Knowledge Extraction algorithms were developed and tested. A prototype associative database was designed.

The associative fact extraction algorithm was implemented in two steps. The first step processes the lexicon. That is, the entire lexicon is input and passed by 64k words of raw text (assuming a 64k data parallel computer). The PEs may be loaded with more than one word apiece to optimize I/O or for other considerations, however, the lexicon processing timing is still the same, as the additional words would appear as virtual PEs and would require a separate lexicon pass for each set of 64k words. During the lexicon pass, all of the text is searched for lexicon element, thus the time is a function of the size of the lexicon. The measured results (not addressing I/O) are 208.7 text words searched per instruction. So a lexicon of 16k (16384) words would require 5,144,576 instructions or .0514 seconds (assuming 10 nano seconds per instruction) to search a 64k text. This is 1273885.35 words of raw text compared against 16384 words of lexicon per second (or 20g comparisons per second vs 0.1g comparisons for a conventional computer of the same speed - a factor of 200 BUT the SIMD number includes overhead, the conventional computer number does not).

		Expert Selected		
Vector	Output	A	B	C
1	Actual		2	1
	Expected		2	1
2	Actual	1	2	
	Expected	1	2	
3	Actual	2		1
	Expected		2	1
4	Actual		1	2
	Expected		1	2
5	Actual		1	2
	Expected	1	2	
6	Actual		1	2
	Expected		1	2
7	Actual	2		1
	Expected	2		1
8	Actual	2		1
	Expected	2		1
9	Actual		2	1
	Expected		2	1
10	Actual	1	2	
	Expected	1	2	
11	Actual	2	1	
	Expected	2	1	
12	Actual		1	2
	Expected		1	2
13	Actual	1		2
	Expected	1		2
14	Actual		2	1
	Expected	2		1
15	Actual		1	2
	Expected		1	2
16	Actual	1		2
	Expected	1		2
17	Actual	2	1	
	Expected	2	1	
18	Actual		1	2
	Expected		1	2
19	Actual	1		2
	Expected	1		2
20	Actual	2	1	
	Expected	2	1	

Table 4.2-1 NNDC Expert Selection

The test data resulted in 0.090 to 0.202 facts extracted per word processed. This is probably a relatively high amount as the test files were chosen to be fact rich. On average, each fact extraction required about 1500 instructions (or 15 microseconds)

The basic design of the NNDC component (Section 3.2.1.2) was tested using input from the AFE component similar to that shown in Figure 3.2.1.3-1. The neural net consisted of 6 input facts (24 nodes) and 3 output nodes (experts). After training, 20 random vectors of 6 facts with random strengths were presented to the network. The output of the network was recorded along with it's associated input vector. The actual output was compared to expected output with the results shown in Table 4.2-1. The neural net assigns probabilities to the 3 experts and the two highest are chosen (noted by 1 and 2 in the Expert Selected columns). The neural net output conformed to the expected output in all but 4 of the total 60 outputs (93.3% success). There was a single failure in vectors 3 and 14 and a double failure in vector 5. The data used was selected strictly for the purpose of conducting tests of this procedure. The expected output was manually selected based on the same criteria that produced the training set.

When the domain experts are notified of a new fact by the NNDC, the Domain Knowledge Extraction Experts review the fact for completeness. The Pronoun expert requires 20 micro seconds of overhead per fact plus 6 microseconds for each word in the pronoun referent. The Not expert requires 1.34 microseconds for each fact (sentence) analyzed. The When expert is implemented similarly to the pronoun expert, it requires 20 microseconds of overhead plus 6 microseconds for each word in the time expression. The timings and algorithms were developed in isolation on a one by one basis, but in actuality, a considerable portion of the overhead would be absorbed by the lexicon processing. Simply put, the lexicon processor would contain the special keywords the experts look for and when found, the lexicon processor marks then with flags. The lexicon processor can perform this effort in basically the same amount of time for the entire 64k word document as the extraction experts require for one fact.

An associative database interface was designed for the SKBAM. The associative database interface lets the SKBAM user access the data and ask questions in a pseudo-natural language format. Facts in the tables generated in response to the table pattern of Figure 3.2-1 can be accessed by the names in the who column. Thus the query "Did China sign the treaty?" will access all facts with China in the who column which will then be further processed looking for treaty in the what column. But all facts may also be accessed associatively, i.e. by "describing" them. For example, "Who did not sign the anti-ballistic missile treaty?" In order to answer this question, all facts containing anti-ballistic missile treaty must be identified, the associated who column extracted and deleted from the list of all the names in the who column. Processing negative questions such as this can be very time consuming. However, the timings for the associative interface were not determined because it is real time. That is, the interface can process the input faster than the relatively slow user can generate it - the SKBAM process input and finds the local answer faster than the user can speak. When the user asks a question that requires anchor system support, the time for a response may be considerable, but beyond the scope of this research.

In order to verify the above timing estimates, which were determined by emulation, and establish the impact of system integration on them, the next phase would require physical implementation and integration of key components on actual hardware such as a Pyxsys SIMD computer. For example, the exact density of facts in the unstructured text will have a great impact on the overall speed of the algorithms. As mentioned above, all of the times are based on a relatively high density. In reality, it is expected that less than 1% of the raw text would have any hits at all.

As mentioned above, data parallel implementation of the neural net component promises very fast execution, but it needs to be verified. The next step would be to implement the neural net on the data parallel computer and test it with actual data in a small, easily verifiable data set. Upon successful completion of this test, a neural net would be constructed for a real domain and verified against actual expert data. As has been described, the neural net chosen is a simple feed forward network. After the procedure in general is verified and found to be sound. The 93.3% success rate can be improved by replacing this simple neural net by a more sophisticated neural network or hybrid network that incorporates a rule-based layer. Another way to increase the success rate could be in the experts themselves. Experts can share data and compare results to verify their decisions. If significantly conflicting or inconsistent data is found between experts they could augment the input and resubmit to the network to produce a new, hopefully better, selection.

Since access to public and military data bases over long distances should be "invisible" to the SKBAM user, it is critical to implement this portion of the system on hardware simulating the actual communication links from SKBAM to anchor to database.

5.0 CONCLUSION

The solution to many future problems depends on an advanced data/knowledge base capability. However, we do not need research on how to build large data/knowledge bases, we need new research on how to make knowledge in knowledge bases useful, how to get it to the user over long distances and from multiple sources, how to discover knowledge in unstructured knowledge bases, how to use semi-structured knowledge, how to access data using "conventional communication skills", how to "integrate" knowledge from diverse sources and how to keep knowledge secure.

Data on a smart field deployable unit suitable for Air Force challenge problems is likely to be fragments of knowledge bases, raw data, complete and incomplete data bases and the user will need all of the above features to use all available knowledge such as 1) semi-structured and unstructured text, 2) WEB based data, 3) existing public databases, 4) existing military databases such as the digital terrain database.

The initial algorithm analysis is promising, but actual test implementation as described in Section 4.0 is required to further this research.

6.0 REFERENCES

1. [II98] Preliminary report of the Intelligent Knowledge Base Systems Working Group of the Information Institute's Workshop 98, June 24-26, 1998, Minnowbrook , NY.

2. [Kozaczynski, 1992] Kozaczynski, Wojtek, Jim Ning and Tom Sarver, "Program Concept Recognition," in Proceedings KBSE '92, McLean, VA., Sept. 20-23, 1992, pp. 216-225.

3. [Brockman, 1997] Brockman, Jay B. and Peter M. Kogge, The Case for Processing-in-Memory, University of Notre Dame, Department of Computer Science and Engineering, TR-97-3.

4. [Keeton, 1997] Keeton, Kimberly, Remzi Arpaci-Dusseau, and David A. Patterson, "IRAM and SmartSIMM: Overcoming the I/O Bus Bottleneck", University of California at Berkeley, Computer Science Division, http://iram.cs.berkeley.edu

5. [Watkins, 1989] Watkins, C., Learning with Delayed Rewards, PhD. Thesis, Cambridge University, Cambridge, UK, 1989.

6.[Masters, 1993] Masters, T., Practical Neural Networks Recipes in C++, Academic Press Inc., 1993, p. 177.

7.[Rosenberg, 1987] Rosenberg, C. and G. Blelloch, "An Implementation of Network Learning on the Connection Machine, " in The Proc. of the Joint Conf. on Artificial Intelligence, Milan, Italy, 1987, pp. 329-340.

8. [Falkoff, 1962] Falkoff, A., "Algorithms for Parallel Search Memories," J. Associative Computing, Mar. 1962, pp. 488-511.

9.[Burleson, 1998] Burleson, D. K. Inside the Database Object Model, CRC Press, Boca Raton, FL., 1998.

10. [Potter, 1992] Potter, J. L., Associate Computing, Plenum Publishing, New York, 1992.

11. [Potter, 1997] Potter, J. L., "ACE: An Associative Calculus Data Structure" in Journal of Parallel and Distributed Computing, No 51, 1998, pp. 63-74.

12.[Schneider, 1998] Schneider, G. and J. Gersting, AN INVITATION TO COMPUTER SCIENCE, PWS Publishing, Pacific Grove, 1996.

13.[Potter, 1994] Potter, J. L., "ASC: An Associative Computing Paradigm," Computer, November, 1994, pp. 19-26.

14.[Bowman, 1997] Bowman, N., N. Cardwell, C. Kozyrakis, C. Romer and H. Wang, "Evaluation of Existing Architectures in IRAM Systems", University of California at Berkeley, Computer Science Division, http://iram.cs.berkeley.edu

15.[Liuzzi, 1980] Liuzzi, R. The Specification of a Data Base Machine Architecture Facility and a Methodology for Developing Special Purpose Function Architectures, RADC-TR-80-256, 1980.

16.[DeFiore, 1971] DeFiore, C., N. Stillman and B. Berra, Associative Techniques in the Solution of Data Management Problems, Proc. ACM '71, August 3-5, 1971, Chicago, Ill, p. 28-36.

17.[Stillman, 1971] Stillman, N., C. DeFiore and P.B. Berra, Associative processing of line drawings, in The Proceedings of the Spring Joint Computer Conference, 1971, pp. 557-562.

18. [Orlando, 1971] Orlando, V. and P. B. Berra, Associative Processors in the Solution of Network Problems, 39th National ORSA Meeting, May 5-7, 1971, Dallas, Texas

A DISTRIBUTED CONCURRENT INTRUSION DETECTION AND RECOVERY SCHEME BASED ON ASSERTIONS

Shambhu J. Upadhyaya
Associate Professor
Department of Computer Science and Engineering

State University of New York at Buffalo
Buffalo, NY 14260

Final Report for:
Summer Research Extension Program (SREP)
(AFOSR Contract Number - F49629-93-C-0063)
Air Force Research Laboratory
Information Directorate
Rome Research Site

Sponsored by:
Air Force Office of Scientific Research
Bolling Air Force Base, DC

and

Air Force Research Laboratory
Information Directorate
Rome Research Site

February 2000

A DISTRIBUTED CONCURRENT INTRUSION DETECTION AND RECOVERY SCHEME BASED ON ASSERTIONS

Shambhu J. Upadhyaya
Associate Professor
Department of Computer Science and Engineering
State University of New York at Buffalo

<u>Abstract</u>

This report describes the development of a new intrusion detection scheme based on concurrent monitoring of user operations. In this scheme, prior to starting a session on a computer, an auxiliary process called a monitor watchdog queries users for a scope file and then generates a table called a sprint-plan. The sprint-plan is composed of carefully derived assertions that can be used as a basis for concurrent monitoring of user commands. The plan is general enough to allow a normal user to perform his task without much interference from the watchdog or system administrator and is specific enough to detect intrusions, both external and internal. Our scheme leverages the concept of signature analysis which has successfully been used for concurrent error detection in the fault tolerance domain.

The funding has resulted in a preliminary prototype of an on-line monitoring tool that can be employed for monitoring intrusions in a distributed computing environment. This scheme is a significant enhancement over the traditional approaches that rely on audit trail analysis in that the intrusion detection is concurrent and does not require the processing of audit data. This can help in fast recovery from successful break-ins. Our effort includes the development of a graphical user interface for the easy user-input and the implementation of the distributed monitoring platform. The research also brings out some new ideas on reasonableness checking for the automatic generation of assertions.

A DISTRIBUTED CONCURRENT INTRUSION DETECTION AND RECOVERY SCHEME BASED ON ASSERTIONS

Shambhu J. Upadhyaya

1 Introduction

Preemption of attack prior to any damage is what is primarily desired of information survivability research. However, it is unlikely that such objectives can ever be met with the current practice of intrusion detection using audit trail analysis. Clearly, audit trail analysis provides only an after-the-fact solution. Therefore, a proactive approach is necessary to detect intrusions as soon as they occur so that damage could be minimized and service be restored without significant downtime [1]. Ability to quickly and correctly detect and identify malicious information attacks is critical to information survivability [2]. In this research, we try to advance a step closer to the said goal by developing novel schemes that do not entirely depend on passive audit trail analysis. Our research leverages two well established concepts from traditional fault tolerance, namely, concurrent monitoring using assertions and reasonableness checking, and uses audit trail analysis techniques as an added layer of protection.

The problem with the approaches based on audit trail analysis is that a vast amount of data is created, especially when user load is high. It is difficult for any detection scheme to catch up with the generation of audit data [3]. Since only a fraction of the data is relevant for intrusion analysis, filtering and/or multi-level processing of audit data may help [4]. However, the approaches using audit data analysis as the base-line technique cannot provide a truly real-time detection because any on-line efforts to reduce the data size inherently slow down the detection process. We present a fundamentally different approach in order to address survivability in high security applications.

In our approach, intrusions are captured on the basis of satisfiability of computing premises by observing conclusions resulting from user actions [5]. This set of assertable conclusions, or simply assertions, is generated by a strategic query of system users at their first login of the daily session, irrespective of whether the subject is a bona fide user or an intruder. The generation of this finite set of assertions by a direct query of users and its processing for intrusion monitoring is based on sound principles of signature analysis [6]. First of all, any inconsistent or anomalous input submitted by the users may raise suspicion and serve as a trigger for intrusion detection. Secondly, our method does not need processing of huge audit data. Hence, detection can occur with low latency and in real-time. Though our technique does not depend on audit trail analysis, it can still be used as a simple add-on third party module to enhance the detection coverage of audit trail analysis based detection schemes.

We consider a distributed system of a network of workstations or PCs as the application domain. A hierarchical monitoring scheme has been developed based on the notion of verifiable assertions. In the remainder of this report, the basic principle of the distributed concurrent intrusion detection scheme and the prototype implementation are given. A detailed description of the architecture, an algorithm and some illustrations appear in [5].

2 Basic Principle

Our concurrent intrusion detection technique is based on the simple principle of satisfiability of computing premises by observing conclusions resulting from user actions. The system starts by first generating a plan for users and then monitoring the users for any significant deviation from this plan. The plan is composed of verifiable *assertions* that are automatically generated on the basis of a few system usage inputs provided at the start of a session. Once an assertable plan called sprint plan (Signature Powered Revised INstruction Table) is generated, the user will be monitored to see how close the plan is maintained during a given session. Certain tolerance limits are set to take care of unplanned, but authentic deviations. Any significant deviations from the plan can be interpreted as an intrusion, external or internal. At this stage, either an interactive dialog is opened with the user or the information is shared with other monitors in the network to initiate system level recovery if the threat is viewed real. A block schematic of the intrusion detection approach with a separation between the one-time effort and the run-time events is given in Figure 1.

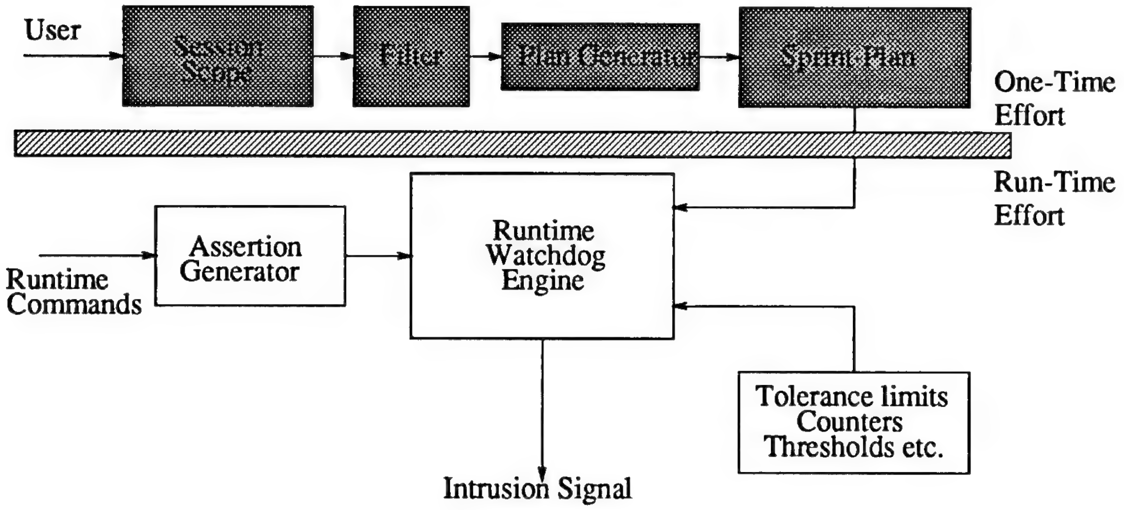


Figure 1: Simplified Flow Diagram of Concurrent Intrusion Detection

Figure 2 shows the block schematic of the monitor process at run-time. Some of the enhancements to reduce false positives are shown in the figure by dotted boxes.

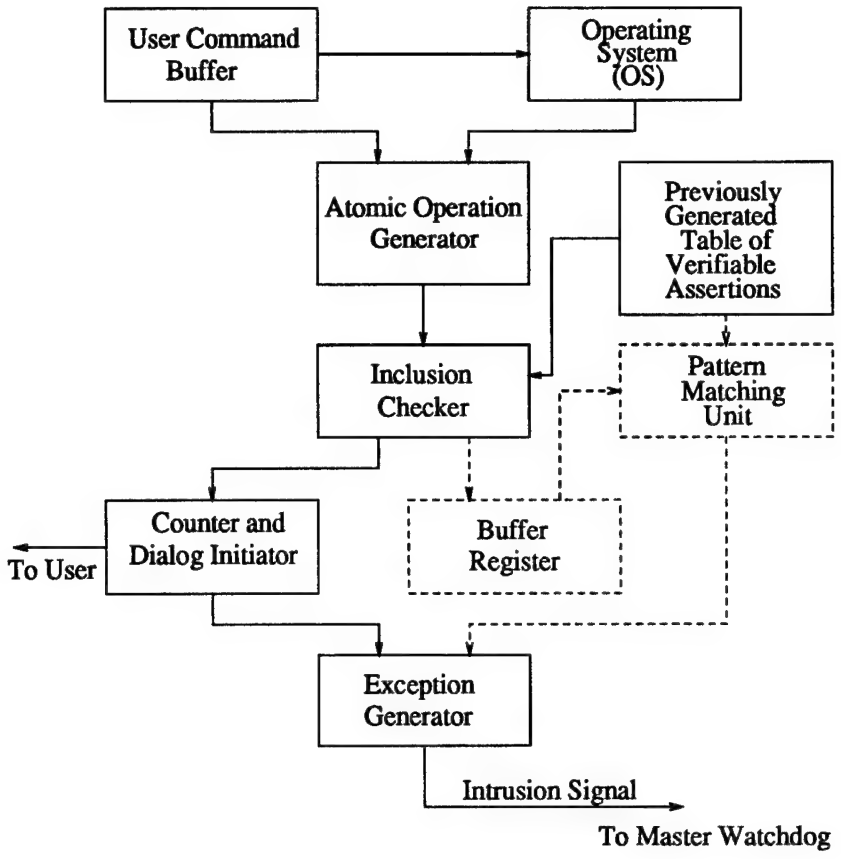


Figure 2: Block Schematic of the Monitor

3 Prototype Development

As described before, the basic system prototype consists of 3 main components, namely, a one-time monitor, a run-time monitor and a comparison module. The one-time monitor has to interact with the users and hence is implemented into a graphical user interface (GUI) using the Java language. The GUI will ease the user input of session scope.

The one-time monitoring takes place as soon as the user logs in. When a user logs in with a userid password submission, the password verification is done first as shown in the flowchart of Figure 3. There is a counter which keeps track of the number of login attempts and the user is given only a specific number of attempts to login. This number is decided upon the amount of security we are looking at for the particular system. If the user is authenticated to login he/she is provided with a series of GUI windows to specify the scope of the session. The user selects a particular application he/she is planning to work on. At this stage of the implementation the user can select only one application at a time and this will be extended so that multiple applications can be selected in a particular session. If, say, the user selects Research as the application, the system provides the user with a preselected input list containing various categories like simulators, design tools, operating

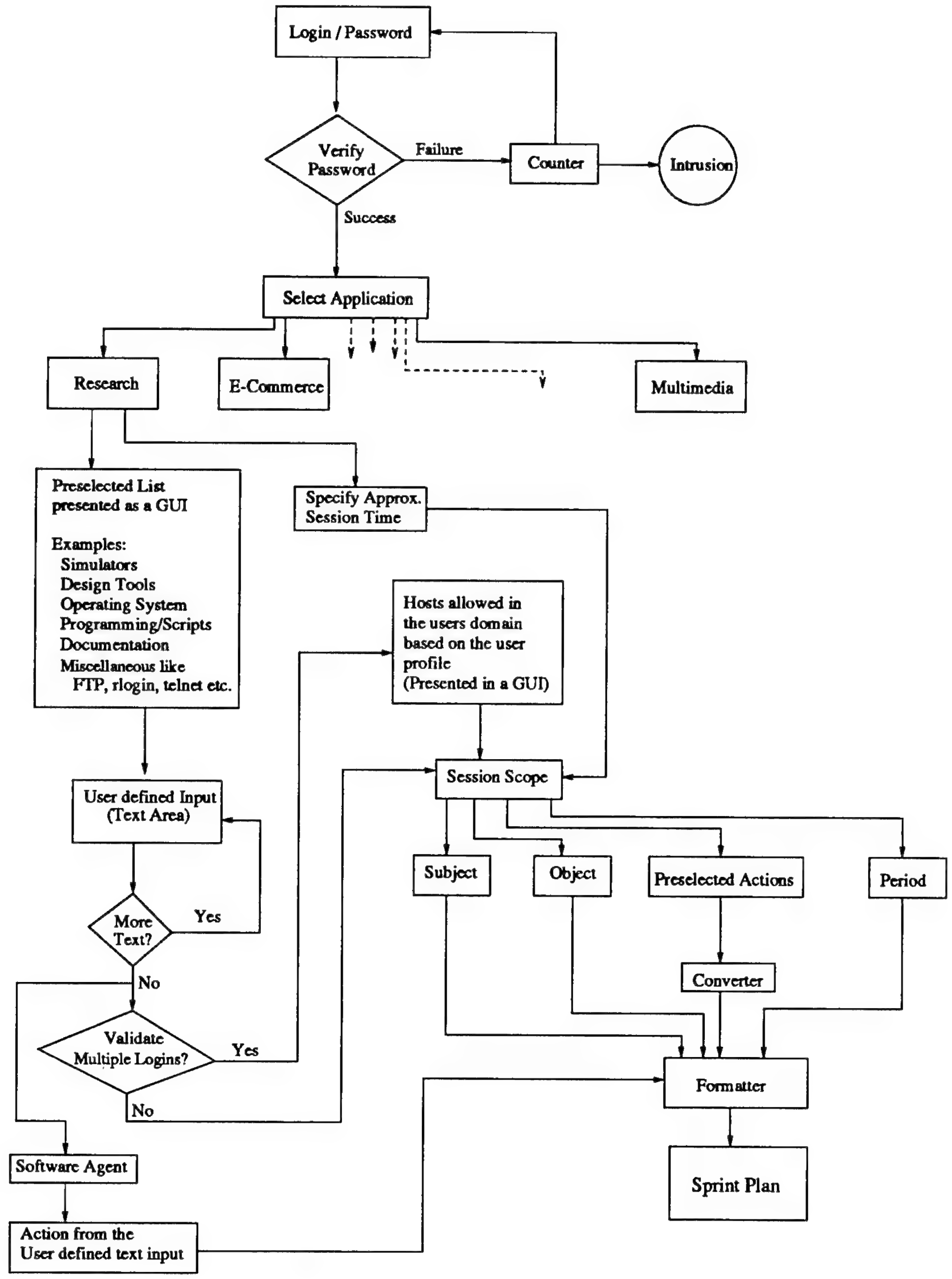


Figure 3: Flowchart of GUI Development

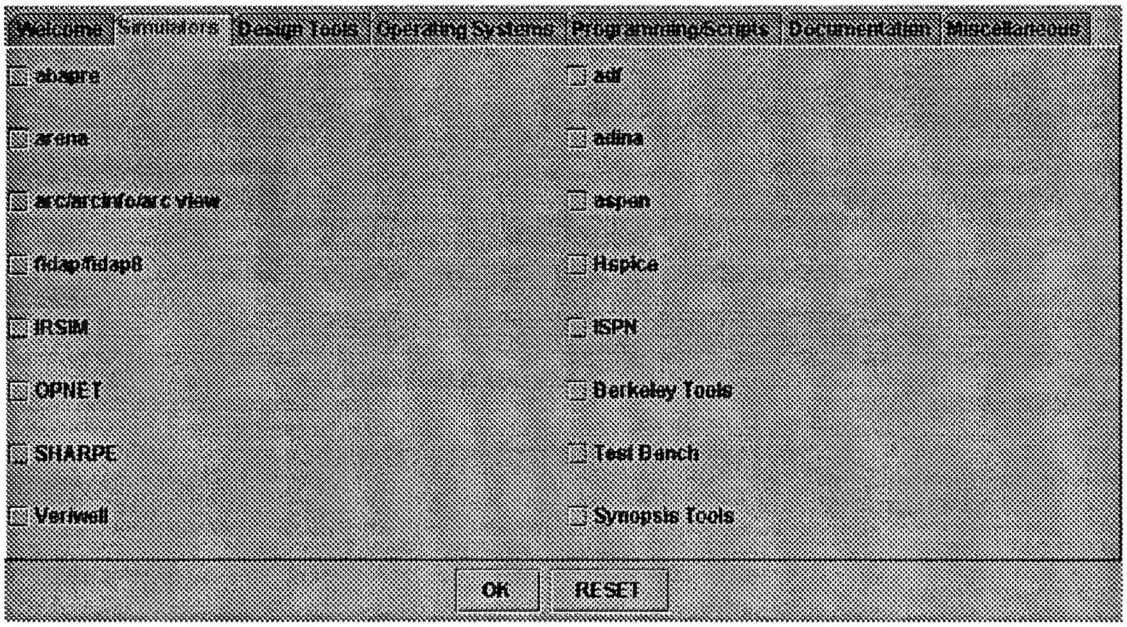


Figure 4: Choice of Simulators on a System

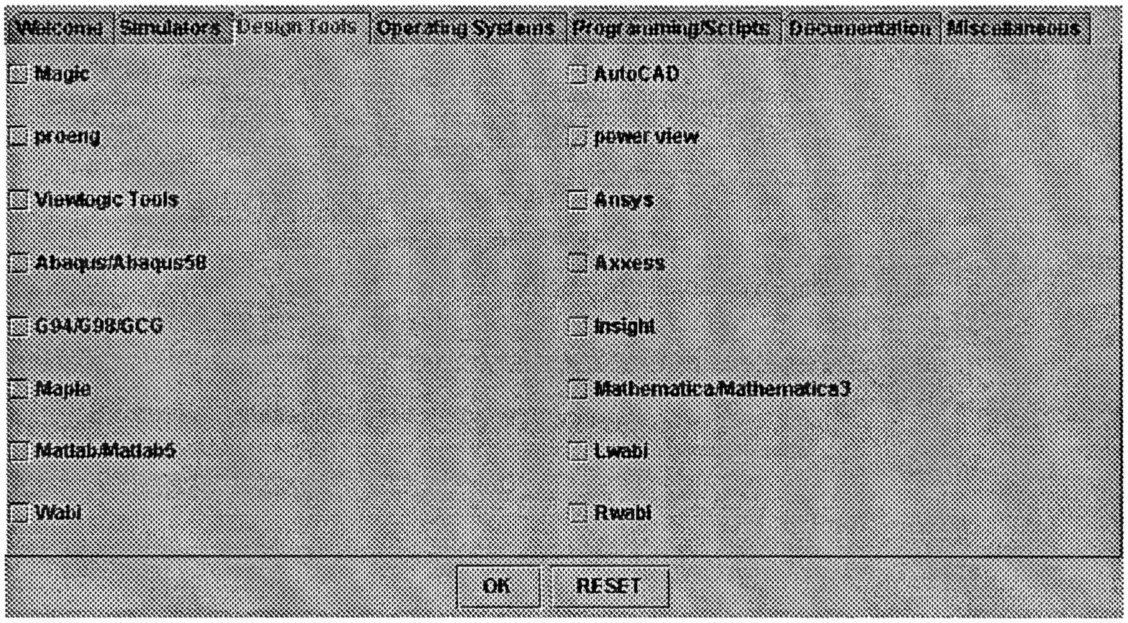


Figure 5: Choice of Design Tools on a System

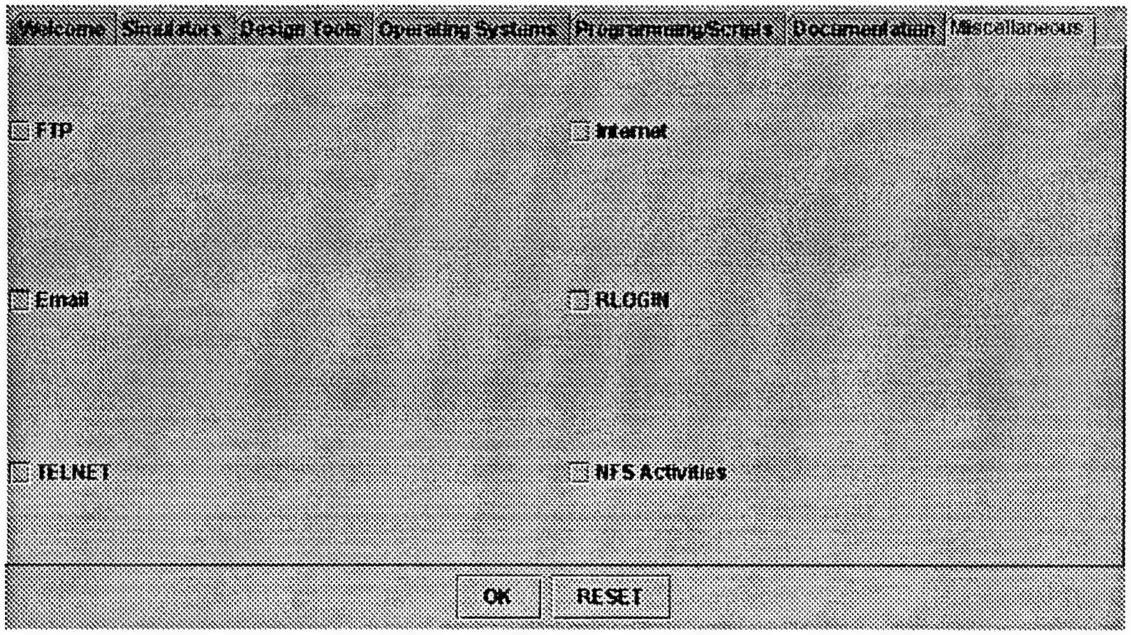
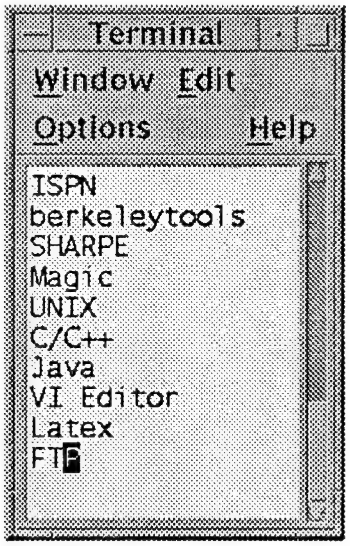
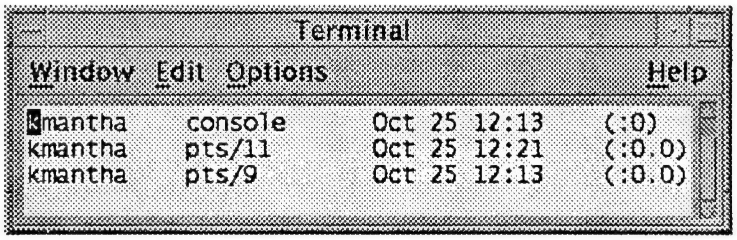


Figure 6: Choice of Miscellaneous Items on a System



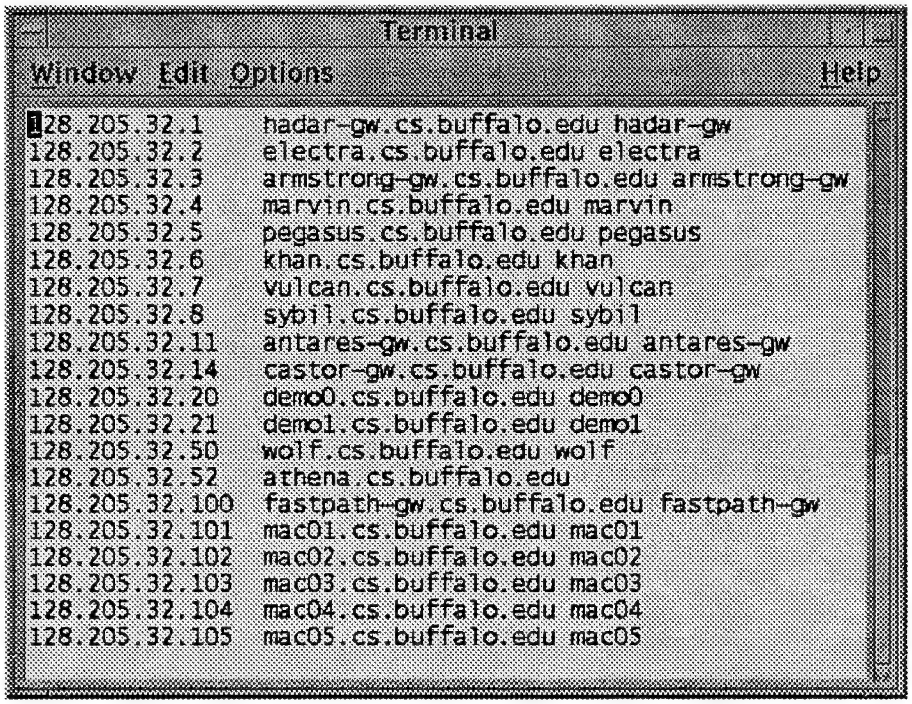
```
ISPN
berkeleytools
SHARPE
Magic
UNIX
C/C++
Java
VI Editor
Latex
FTP
```

Figure 7: An Example Session Scope



```
kmantha    console     Oct 25 12:13    (:0)
kmantha    pts/11      Oct 25 12:21    (:0.0)
kmantha    pts/9       Oct 25 12:13    (:0.0)
```

Figure 8: Login Information Generated by the Converter



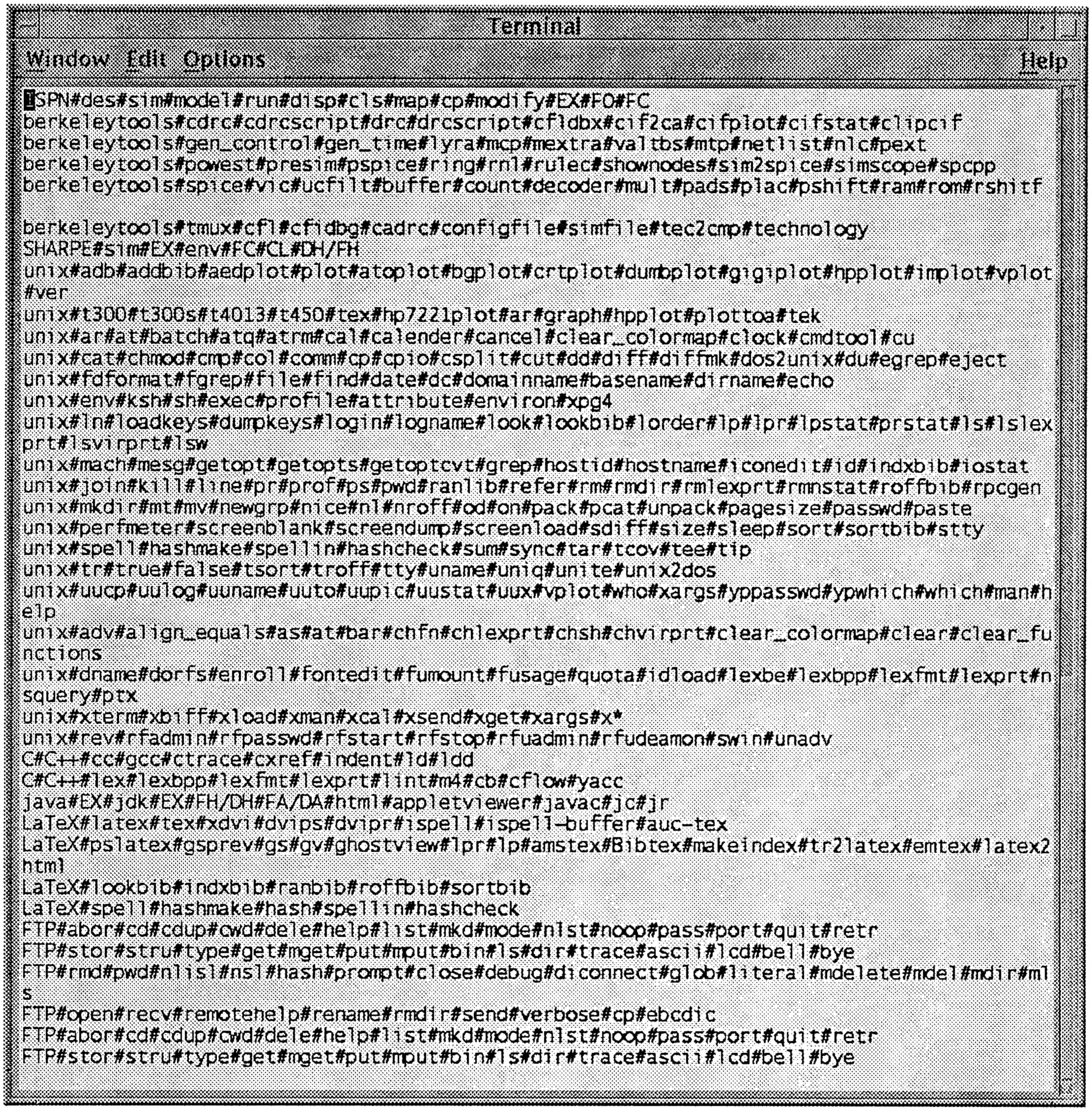
```
128.205.32.1     hadar-gw.cs.buffalo.edu hadar-gw
128.205.32.2     electra.cs.buffalo.edu electra
128.205.32.3     armstrong-gw.cs.buffalo.edu armstrong-gw
128.205.32.4     marvin.cs.buffalo.edu marvin
128.205.32.5     pegasus.cs.buffalo.edu pegasus
128.205.32.6     khan.cs.buffalo.edu khan
128.205.32.7     vulcan.cs.buffalo.edu vulcan
128.205.32.8     sybil.cs.buffalo.edu sybil
128.205.32.11    antares-gw.cs.buffalo.edu antares-gw
128.205.32.14    castor-gw.cs.buffalo.edu castor-gw
128.205.32.20    demo0.cs.buffalo.edu demo0
128.205.32.21    demo1.cs.buffalo.edu demo1
128.205.32.50    wolf.cs.buffalo.edu wolf
128.205.32.52    athena.cs.buffalo.edu
128.205.32.100   fastpath-gw.cs.buffalo.edu fastpath-gw
128.205.32.101   mac01.cs.buffalo.edu mac01
128.205.32.102   mac02.cs.buffalo.edu mac02
128.205.32.103   mac03.cs.buffalo.edu mac03
128.205.32.104   mac04.cs.buffalo.edu mac04
128.205.32.105   mac05.cs.buffalo.edu mac05
```

Figure 9: List of Allowed Hosts Generated by the Converter

system, programming languages, scripts, documentation and miscellaneous items like ftp, r-login, telnet etc. Figures 4, 5, 6 show the screen images of the GUI illustrating various choices of Simulators, Design Tools and Miscellaneous items respectively.

The user just needs to check off the tasks he/she intends to perform. Once this is done, the watchdog queries the user if he/she intends to specify any other items that were not present in the predetermined list. This is achieved by entering text in the text area provided by the watchdog. Then the user is queried if he intends to open multiple sessions. If the user selects Yes, a preselected list of all hosts in that particular domain which the user is authenticated to connect, is given based on his profile. The user can check off all the places he intends to connect to. Then the watchdog subdivides the information obtained into 5 parts, subject (users login information), object (objects the user is going to modify in the session), preselected actions (obtained from the preselected input), text actions (obtained from the text area), and period (approximate time of the session). The preselected list of actions is then processed by a converter to produce a sprint plan. The text action part is fed to a software agent. This agent is an autonomous intelligent entity and is responsible to understand the text input given by the user and convert it into the sprint plan. Figure 7 gives a summary of the session scope generated by the GUI based on some user input.

The various components of the sprint plan are shown in Figures 8, 9, 10. These components are combined together by a formatter to obtain the final sprint plan (not shown). This completes the one-time monitoring part. The implementation of this as well as the run-time



```
SPN#des#sim#model#run#disp#cls#map#cp#modify#EX#FO#FC
berkeleytools#cdrc#cdrcscript#drc#drcscript#cfldbx#cif2ca#cifplot#cifstat#clipcif
berkeleytools#gen_control#gen_time#lyra#mcp#mextra#valtbs#mtp#netlist#nlc#pext
berkeleytools#powest#presim#pspice#ring#rnl#rulec#shownodes#sim2spice#simscope#spcpp
berkeleytools#spice#vic#ucfilt#buffer#count#decoder#mult#pads#plac#pshift#ram#rom#rshitf

berkeleytools#tmux#cfl#cfidbg#cadrc#configfile#simfile#tec2cmp#technology
SHARPE#sim#EX#env#FC#CL#DH/FH
unix#adb#addbib#aedplot#plot#atoplot#bgplot#crtplot#dumbplot#gigiplot#hpplot#implot#vplot
#ver
unix#t300#t300s#t4013#t450#tex#hp7221plot#ar#graph#hpplot#plottoa#tek
unix#ar#at#batch#atq#atrm#cal#calender#cancel#clear_colormap#clock#cmdtool#cu
unix#cat#chmod#cmp#col#comm#cp#cpio#csplit#cut#dd#diff#diffmk#dos2unix#du#egrep#eject
unix#fdformat#fgrep#file#find#date#dc#domainname#basename#dirname#echo
unix#env#ksh#sh#exec#profile#attribute#environ#xpg4
unix#ln#loadkeys#dumpkeys#login#logname#look#lookbib#lorder#lp#lpr#lpstat#prstat#ls#lslex
prt#lsvirprt#lsw
unix#mach#mesg#getopt#getopts#getoptcvt#grep#hostid#hostname#iconedit#id#indxbib#iostat
unix#join#kill#line#pr#prof#ps#pwd#ranlib#refer#rm#rmdir#rmlexprt#rmnstat#roffbib#rpcgen
unix#mkdir#mt#mv#newgrp#nice#nl#nroff#od#on#pack#pcat#unpack#pagesize#passwd#paste
unix#perfmeter#screenblank#screendump#screenload#sdiff#size#sleep#sort#sortbib#stty
unix#spell#hashmake#spellin#hashcheck#sum#sync#tar#tcov#tee#tip
unix#tr#true#false#tsort#troff#tty#uname#uniq#unite#unix2dos
unix#uucp#uulog#uuname#uuto#uupic#uustat#uux#vplot#who#xargs#yppasswd#ypwhich#which#man#h
elp
unix#adv#align_equals#as#at#bar#chfn#chlexprt#chsh#chvirprt#clear_colormap#clear#clear_fu
nctions
unix#dname#dorfs#enroll#fontedit#fumount#fusage#quota#idload#lexbe#lexbpp#lexfmt#lexprt#n
squery#ptx
unix#xterm#xbiff#xload#xman#xcal#xsend#xget#xargs#x*
unix#rev#rfadmin#rfpasswd#rfstart#rfstop#rfuadmin#rfudeamon#swin#unadv
C#C++#cc#gcc#ctrace#cxref#indent#ld#ldd
C#C++#lex#lexbpp#lexfmt#lexprt#lint#m4#cb#cflow#yacc
java#EX#jdk#EX#FH/DH#FA/DA#html#appletviewer#javac#jc#jr
LaTeX#latex#tex#xdvi#dvips#dvipr#ispell#ispell-buffer#auc-tex
LaTeX#pslatex#gsprev#gs#gv#ghostview#lpr#lp#amstex#Bibtex#makeindex#tr2latex#emtex#latex2
html
LaTeX#lookbib#indxbib#ranbib#roffbib#sortbib
LaTeX#spell#hashmake#hash#spellin#hashcheck
FTP#abor#cd#cdup#cwd#dele#help#list#mkd#mode#nlst#noop#pass#port#quit#retr
FTP#stor#stru#type#get#mget#put#mput#bin#ls#dir#trace#ascii#lcd#bell#bye
FTP#rmd#pwd#nlisl#nsl#hash#prompt#close#debug#diconnect#glob#literal#mdelete#mdel#mdir#ml
s
FTP#open#recv#remotehelp#rename#rmdir#send#verbose#cp#ebcdic
FTP#abor#cd#cdup#cwd#dele#help#list#mkd#mode#nlst#noop#pass#port#quit#retr
FTP#stor#stru#type#get#mget#put#mput#bin#ls#dir#trace#ascii#lcd#bell#bye
```

Figure 10: The 'Action' Component of Sprint Plan Generated by the Converter

monitoring and the comparison part also have been successfully completed. The entire GUI was implemented using a platform neutral language such as Java.

4 Architecture Simulation

The simulation setup is Java-based. To keep the complexity of implementation to a minimum, the system is initially simulated for one user and two hosts and a server. The auxiliary monitor watchdog is also simulated. This watchdog queries a user as soon as he/she logs on. The query is initiated with the help of a graphical user interface. The inputs to the query are stored in a scope-file which will be used to generate the sprint plan. If the current login is a second session, then the system will not query the user for the scope file. Instead, the already generated sprint plan will be used from the server for monitoring purposes. The watchdog continuously monitors the user activity, compares the run-time inputs to the sprint plan, thereby generating exceptions with the help of a dialog initiator and a set of safety counters. The session-scope captured from the user and the sprint plan are stored in a secure place in the server. The run-time monitoring of the user activity is hidden from the user. However, the user can update the scope file upon submitting a request to the watchdog. An intelligent software agents assists the watchdog in converting the user defined input of the scope-file into the appropriate assertions in the sprint plan. The architecture of the simulated system is shown in Figure 11. The software agent and the file watchdog which is responsible to guard the secure files in the server are not fully implemented at this stage.

5 New Ideas and and Future Work

In order to establish the generality of our work, a commercial (virtual) banking environment is considered. Here, attention is given to simulating the employees in a bank rather than the customer so that we can test for misuse intrusions. An employee in a bank might have access to sensitive data and there is always a possibility of misusing the previleges. The simulated banking environment supports multiple banks with multiple accounts with multiple employees in the bank having access to all the accounts. The customer is also simulated to some extent allowing online access to checking account. The customer is also provided with ATM facility. The dialog between the customer and the employee is being simulated. Further details of this simulation will appear in [7].

The basic prototype and application environment simulation will facilitate intrusion injection experiments and the evaluation of our concurrent intrusion detection technique. The actual evaluation was considered only as a long term objective and was outside the scope of this project. Several new ideas have been developed to improve and automate the sprint plan conversion process. One such idea is the development of a reasonableness check framework. Techniques such as computational thread following, and temporal and structural sequence

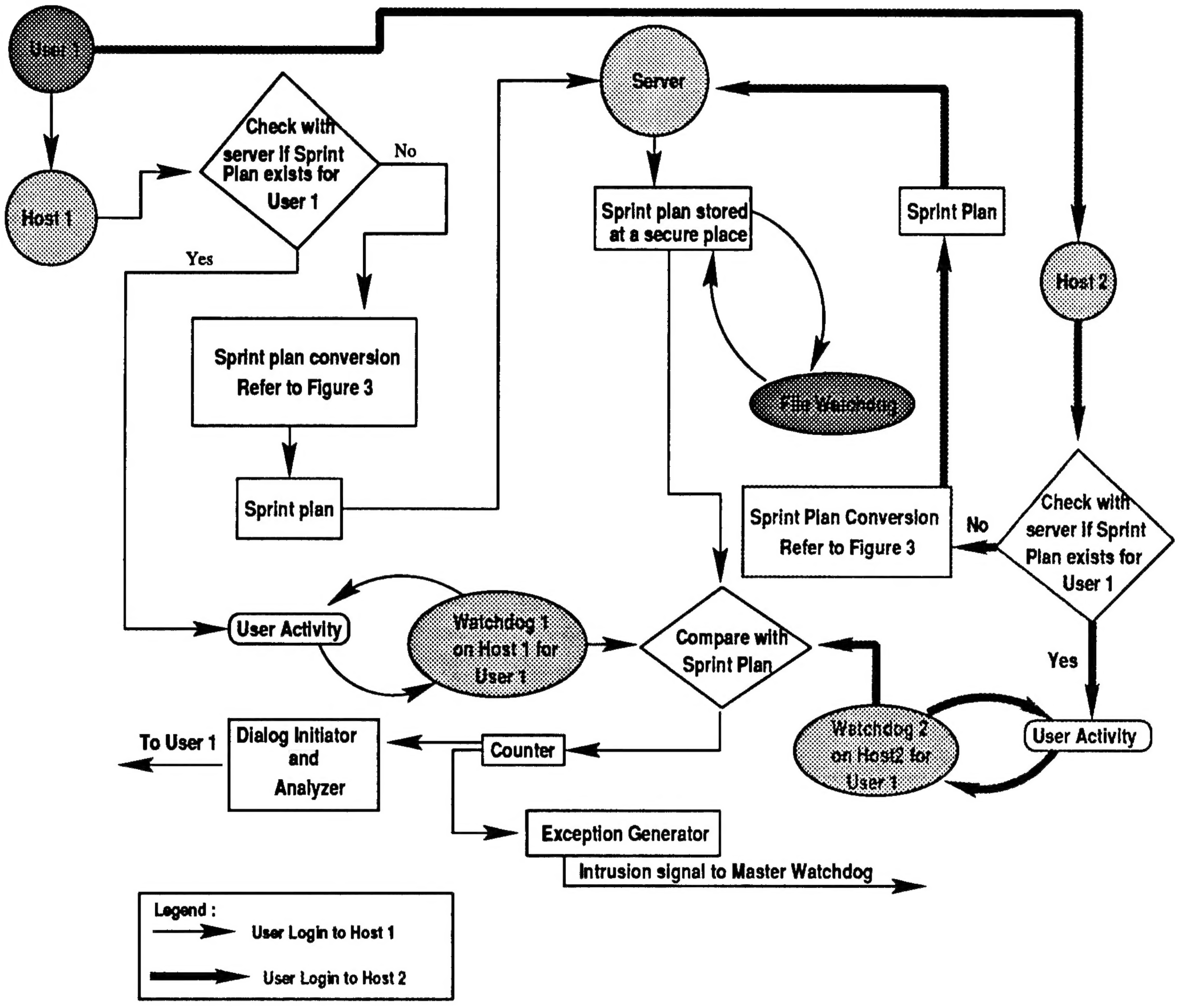

User 1
Host 1
Check with server if Sprint Plan exists for User 1
No
Yes
Sprint plan conversion Refer to Figure 3
Sprint plan
Server
Sprint plan stored at a secure place
File Watchdog
Sprint Plan
Host 2
Sprint Plan Conversion Refer to Figure 3
No
Check with server if Sprint Plan exists for User 1
Yes
User Activity
Watchdog 1 on Host 1 for User 1
Compare with Sprint Plan
Watchdog 2 on Host2 for User 1
User Activity
To User 1
Dialog Initiator and Analyzer
Counter
Exception Generator
Intrusion signal to Master Watchdog
Legend :
User Login to Host 1
User Login to Host 2

Figure 11: Run-time Monitoring Setup for 1-user, 2-hosts System on a Single Server

monitoring will be used where applicable, in addition to traditional inverse operation testing. The practices used in business and commerce to arrive at reasonable contract pricing [8] and other mathematical estimation techniques will be leveraged in deriving our framework. This framework can be used to determine what is a reasonable plan when a user initially porovides his/her computing premises to the monitor watchdog.

References

[1] J. Feldman, J. Giordano, and J. Palmer, "Information survivability at Rome laboratory," *1997 IEEE Information Survivability Workshop*, 1997.

[2] B. Panda and J. Giordano, "Defensive information warfare: Guest editorial," *Communications of the ACM*, pp. 31–32, July 1999.

[3] K. Ilgun, R. Kemmerer, and P. Porras, "State transition analysis: A rule-based intrusion detection approach," *IEEE Trans. on Software Eng.*, vol. 21, pp. 181–199, March 1995.

[4] P. Porras and P. Neumann, "EMERALD: Event monitoring enabling responses to anomalous live disturbances," *National Information Systems Security Conf.*, pp. 353–365, Oct. 1997.

[5] S. Upadhyaya and K. Kwiat, "A distributed concurrent intrusion detection scheme based on assertions," *SCS International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, pp. 369–376, July 1999.

[6] M. Namjoo, "Techniques for concurrent testing of VLSI processor operation," *Proc. International Test Conference*, pp. 461–468, November 1982.

[7] K. Mantha, R. Chinchani, S. Upadhyaya, and K. Kwiat, "Simulation of intrusion detection in distributed systems," *submitted to SCS Summer Simulation Conference*, July 2000.

[8] U. S. Congress, "DOD contract pricing: Hearing before a subcommittee of the committee on government operations, House of Representatives, One Hundredth Congress, second session," *Washington, U.S. G.P.O*, September 1988.